NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

STEREO IMAGE CORRESPONDENCE METHODS FOR EFFICIENT HARDWARE IMPLEMENTATION

by

Bradley J. Ulis

June 2010

Thesis Advisor:	Roberto Cristi
Second Reader:	Monique P. Fargues

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2010	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE Stereo Image Correspondence Methods for Efficient Hardware Implementation	**5. FUNDING NUMBERS**
6. AUTHOR(S) Bradley J. Ulis	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol Number: _____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

In this thesis, depth information extraction via ordinal measures for logical pattern correlation is investigated, along with matching techniques that translate well into hardware implementations. First, the ordinal correspondence metric is evaluated for complexity and performance using traditional correspondence matching techniques. The results show that the ordinal measures are very robust in stereo correspondence when paired with modest error handling techniques. Second, the ordinal measures are applied to an efficient dynamic programming matching algorithm to produce high-quality disparity maps. It is shown that ordinal measures are demonstrably fast in software and that they can be adapted to an extremely fast hardware implementation to produce high quality disparity maps for very high resolution images in real time.

14. SUBJECT TERMS Stereo Correspondence, Ordinal Measures, 3D Reconstruction, Dynamic Programming	**15. NUMBER OF PAGES** 97
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

STEREO IMAGE CORRESPONDENCE METHODS FOR EFFICIENT HARDWARE IMPLEMENTATION

Bradley J. Ulis
Ensign, United States Navy
B.S., United States Naval Academy, May 2009

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2010

Author: Bradley J. Ulis

Approved by: Roberto Cristi
Thesis Advisor

Monique P. Fargues
Second Reader

R. Clark Robertson
Chairman, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

In this thesis, depth information extraction via ordinal measures for logical pattern correlation is investigated, along with matching techniques that translate well into hardware implementations. First, the ordinal correspondence metric is evaluated for complexity and performance using traditional correspondence matching techniques. The results show that the ordinal measures are very robust in stereo correspondence when paired with modest error handling techniques. Second, the ordinal measures are applied to an efficient dynamic programming matching algorithm to produce high-quality disparity maps. It is shown that ordinal measures are demonstrably fast in software and that they can be adapted to an extremely fast hardware implementation to produce high quality disparity maps for very high resolution images in real time.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Solving the stereo correspondence problem to produce disparity maps offers robots an effective means to segment and understand the real world. Disparity maps are a three-dimensional (3D) reconstruction of the scene in front of a stereo pair of cameras. Significant research has gone to make solving for the disparity map faster and more accurate so that they can be implemented in real time for robotics and other commercial applications. A robot capable of determining objects based on their spatial information in a disparity map can potentially interact with them in much the same way humans see and interact with objects.

The objective of this research was to investigate a matching metric and computer vision algorithm, each with acceptable accuracy and potential for efficient hardware implementation. The areas of interest are:

1. The correlation coefficient κ's accuracy in determining matches,
2. The implemented metric hardware utilization and latency.

Also, the correspondence matching algorithms were looked at for:

1. Computational complexity,
2. Accuracy in determining depth from a stereo pair of images,
3. Memory requirements.

For this research, the ordinal correlation coefficient κ was evaluated by the metric objectives, and a traditional pattern matching algorithm and a dynamic programming pattern matching algorithm were evaluated under the correspondence matching algorithm objectives.

The stereo correspondence problem is solved by matching points between a stereo pair of images. For every point in the reference image of the stereo pair, a matching point is found in the target image of the stereo pair. This match is described by a translation vector from the reference pixel in the reference image to the target pixel in the target image. All the disparity vectors consolidate into a dense map called a disparity map. The

disparity map is represented such that the intensity at each point is proportional to the magnitude of the disparity vector (proportional to depth) at each point in the dense map.

The ordinal itself was investigated for its candidacy toward hardware implementation. The proposed architecture can be fully synthesized in a field programmable gate array (FPGA) and computes the first correlation coefficient κ in $5+\lceil \log_2 n \rceil$ clock cycles fully pipelined. While an actual implementation was not possible for this research due to time constraints, an implementation of only the ordinal as a macro function could be sufficient to achieve real-time dense disparity map computation. The ordinal satisfactorily met the objectives of performance and for candidacy toward hardware implementation.

The ordinal coefficient κ was first evaluated for performance with a traditional back-matching strategy. The back-matching strategy is a "greedy" method where the best match within a set of possible candidate matches is chosen by maximizing the correlation coefficient κ. However, the traditional back-matching strategy alone introduces too much error from the matching process as a result of distortions inherent to stereo vision (such as occlusions, specularity, depth boundaries or projective distortion). To mitigate this, the method was improved by ad-hoc error correction. Two such error correction techniques were evaluated: "masked averaged hole fill," resembling a zero-order hold error correction, and "masked linear hole fill," resembling a first order linear interpolation error correction. The results were of significantly higher quality after error correction.

The ordinal coefficient κ was next evaluated using dynamic programming, producing raw results of much higher quality and accuracy compared to the traditional strategy unaided by ad-hoc error correction. The dynamic programming algorithm investigated offered greater latitude in computational complexity and memory usage optimizations over the traditional technique. The memory requirements were proportional to the computational complexity such that an efficient and fast implementation of the dynamic programming method could be achieved in an FPGA. The dynamic programming method met the objectives of computational complexity, memory requirements and accuracy.

ACKNOWLEDGMENTS

I thank my Mother and Father for supporting me in my endeavors and listening to my ceaseless monologue about my research interests.

I thank the United States Navy and everyone involved in giving me an opportunity at an education. I could not have accomplished what I have any other way.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

As the field of robotics advances, powerful sensor fusion and processing is needed to recognize and interact with objects and the world autonomously in real time [1]. There are many avenues for object recognition in robotics such as radar, sonar, optics and manipulation [2], [3], [4]. In the natural world, vision provides the majority of spatial awareness and object recognition since vision and video are signals carrying vast quantities of information about the world. Visual information can be passively obtained from cameras, offering improved security, safety and power advantages for robots. Thus, image and video processing has been an area of interest for advanced robotic intelligence navigation.

Video processing can be computationally intensive, requiring powerful and costly visual systems to process large data streams. Efficient implementation of video processing algorithms is necessary to handle the data meaningfully. Furthermore, accurate object segmentation often requires sophisticated algorithms that can be difficult to implement in real time [5]. An efficient implementation of computer vision algorithms with the intent for use in robotics and machine intelligence was investigated in this research.

1. Computational Complexity

Computer algorithms are analyzed using a special notation that permits a generalized comparison independent of exact implementation details. The notation used in this thesis for algorithm analysis is the *big theta* notation $\Theta(g)$ where g is a function proportional to the average execution time of an algorithm but not approaching asymptotic execution time. The big theta notation accounts for an algorithm's computational complexity (*for* loops and some lower bound constant multipliers) but ignores implementation specific details such as the time needed to execute a particular

procedure such as adding, multiplying or moving data. For example, an algorithm has two *for* loops iterating through a data set that has n input elements. One of the *for* loops also requires a lower bound constant multiplier c to execute. The big theta notation for this algorithm's complexity would be $\Theta\left(cn^2\right)$ indicating that the algorithm would require at least approximately cn^2 time to execute. Also, the big theta notation allows for implicit comparative execution time such that $\Theta\left(m+n\right)=\Theta\left(\max\left\{m,n\right\}\right)$. In this case, the execution time (computational complexity) of the algorithm is at least proportional to the greatest of all the terms in g. The notation can also be used to indicate complexity in memory requirements or complexity in hardware implementation [6].

Hardware implementations can be tailored to be fully sequential or fully parallel. Thus, the computational complexity $\Theta\left(g\right)$ in time comes at a trade-off with the hardware complexity $\Theta\left(f\right)$ in instance replication. Given an algorithm with the potential for full parallel implementation that takes $\Theta\left(n\right)$ time to execute in software, a hardware implementation could instantiate a single instance of the procedure such that it has $\Theta\left(1\right)$ instantiated (spatial) complexity but requires $\Theta\left(n\right)$ time to execute the algorithm. The hardware could also have n instantiations of the algorithm procedure such that it has $\Theta\left(n\right)$ complexity in space but executes that algorithm in $\Theta\left(1\right)$ time. Thus, a machine could have a theoretical lower bound instantiated complexity $\Theta\left(1\right)$ and a theoretical upper bound in instantiated complexity of an algorithms worst case execution time $\Theta\left(n\right)$. This is analogous to transforming an algorithm's execution in time to an equivalent execution in parallel space.

2. Hardware Implementation

Complex procedures can be implemented in hardware as macro functions that can outperform similar software implementations. For example, a floating point multiplication can be accomplished with fixed point multipliers and a software driver, but at reduced performance and power efficiency with respect to a floating point macro in

hardware. For this reason, processors commonly have hardware specific for floating point operations. In many cases, functions and procedures have opportunities for efficient hardware implementation that can speed up the execution of algorithms dependent on such functions.

In computer vision, algorithms commonly iterate through all the pixels of an input image. This gives computer vision algorithms the characteristic burden of dimensionality with at least the growth function $\Theta(MN)$ where M and N represent the number of rows and columns of the image, respectively [7], [8]. Such a growth function can get out of hand quickly, especially if the core procedure is not trivial. For example, if the algorithm were to perform only a single floating point calculation, a slow floating point implementation that executes in t time grows to a total execution time of at least MNt. If t is large and the image has many pixels, a real time execution may not be possible with the slow procedure. In the aforementioned growth function, a procedure is executed *for* every row and *for* every column in each row (nested loops). If the procedure of the inner loop (the loop executed the number of times described by the growth function) is not trivial as commonly seen in computer vision algorithms, a hardware implementation may be necessary for feasibility.

Hardware implementation can be achieved through a wide range of products as all integrated circuits are essentially a hardware implementation of some function or procedure. A widely available device for rapid prototyping and hardware implementation is the field programmable gate array (FPGA). FPGAs are a “blank slate” integrated circuit that can be configured with a hardware function *after* the device has been fabricated. This has made FPGAs useful for assisting algorithms with fast implementations of complex procedures that are too slow for software without the costly design and fabrication of application specific integrated circuits (ASICs).

For this research, hardware implementation is assumed to be implemented in a typical FPGA, such as Altera, which has unique logic synthesis characteristics which may be dramatically different from another FPGA technology. The implementations are intended to be synthesized and run at a clock speed of the order of 100 MHz. Thus, one

clock cycle is defined as 10 nanoseconds, and a circuit that requires one clock cycle in an Altera FPGA (i.e., a Cyclone or Stratix device) must not have any transient activity (clock slack) exceeding 10 nanoseconds. When any architecture is proposed, one must carefully consider the gate-to-gate propagation delay or critical path, which can be highly dependent on a device's architecture and the combinatorial complexity of the circuit to be synthesized. The clock cycle numbers proposed in this research were verified using the Altera Quartus II 8.1 software and the Timequest Timing Analyzer slow 1100mV and 85 degrees Celsius model on the Cyclone family of FPGAs.

3. Software Simulation

Hardware and software implementations are far from exclusive of one another. Algorithms with potential for hardware implementation often still require management of memory and states; also, software implementations of algorithms must always operate above at least some hardware. However, some software routines such as dynamic memory allocation can be very difficult to implement in hardware. Thus, some software routines are not easily synthesizable into a hardware equivalent, but any hardware routine can be simulated with software and basic data management capabilities. An illustration of synthesizable hardware routines and software routines is shown in Figure 1.

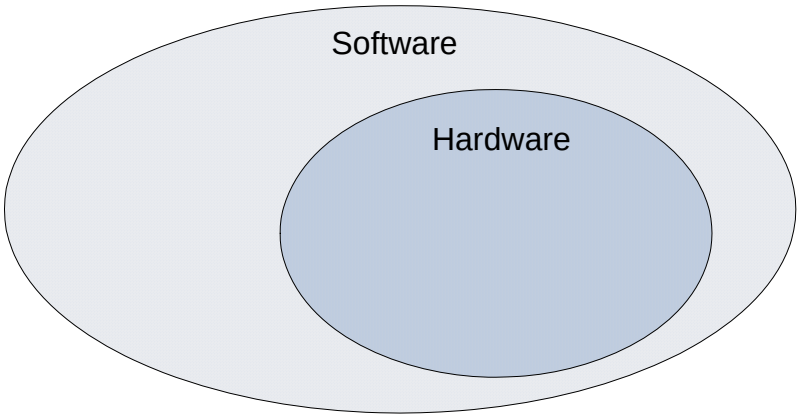


Figure 1. Software and Hardware Venn Diagram.

On a personal computer, software has access to the fundamental hardware building blocks, such as adding and multiplying, needed to construct complex procedures

that may or may not be possible in hardware. For the purpose of this research, software procedures that have potential for efficient hardware implementation are considered but are not actually synthesized. Software simulation of such procedures allows for accurate estimation of hardware performance and complexity.

B. THE CORRESPONDENCE PROBLEM

In computer vision, the depth of a scene can be calculated by solving the correspondence problem between two images. A scene's depth is a three-dimensional process that can be calculated from the distortion between two or more images. A computer must first identify points in the different views that correspond in space, a non-trivial problem that requires complicated pattern matching metrics and schemes to accurately correlate points. Recent research in the correspondence problem has been in algorithms that can accurately identify matching points in real time [5]. Real time is the equivalent of human perception of time or approximately 30 frames per second. The commercial application of the correspondence problem is the generation of disparity maps for object recognition and segmentation. The magnitude of the disparity map is inversely proportional to the depth of the scene represented by the different viewing angles.

Disparity maps are typically a dense matrix of vectors wherein the matrix dimensions are approximately equal to the dimensions of the original images. In a stereo pair of images, one image is taken as the reference image, and the other image is the target image. A pixel from the reference image represents a point in space and is correlated to a pixel in the target image. The horizontal and vertical translation of the correlated point from its reference pixel in the reference image to the target pixel in the target image is described by a vector. This vector is illustrated in Figure 2. The magnitude of each vector in the disparity map is the inverse of the depth at that point in the reference image.

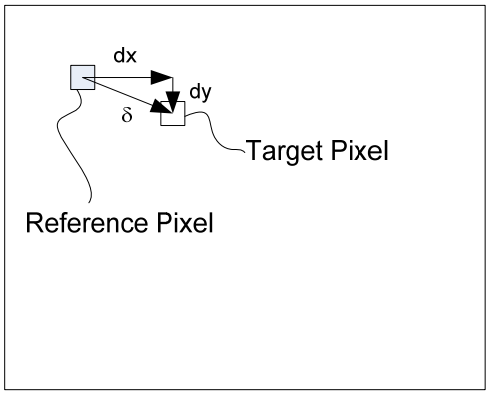


Figure 2. Disparity Map Vector.

A disparity map typically is shown as an intensity image where the intensity at each pixel is the magnitude of the disparity vector. Since disparity is inversely proportional to depth, greater intensity means the correlated point in space is closer to the viewer, and lesser intensity is farther from the viewer. The disparity map can be an accurate representation of objects in space and can be used effectively for object segmentation [9].

The disparity map measures the depth process, and thus, the process of evaluating disparity has an associated *accuracy*. Accuracy in a disparity map is the disparity map's fidelity to the depth process and *not* the appearance of the disparity map itself. Thus, an accurate disparity map will closely sample the depth process. For this research, none of the experiments have a ground-truth depth for objective accuracy evaluation. Therefore, the accuracy of the disparity map is *subjectively* determined by comparing object edges in the disparity map with the corresponding object edges in the reference image manually. If the edges agree, the disparity map is accurate, and if they do not agree, the disparity map is not accurate.

C. OBJECTIVE

The objective of this research was to investigate a matching metric and computer vision algorithm each with acceptable accuracy and potential for efficient hardware implementation. The objectives of interest are:

1. The correlation coefficient κ's accuracy in determining matches,
2. The implemented metric hardware utilization and latency.

Also, the correspondence matching algorithms were looked at for:

1. Computational complexity,
2. Accuracy in determining depth from a stereo pair of images,
3. Memory requirements.

For this research, the ordinal correlation coefficient κ was evaluated by the metric objectives (given a particular matching algorithm), and a traditional pattern matching algorithm and a dynamic programming pattern matching algorithm were evaluated under the correspondence matching algorithm objectives.

D. C/C++ IMPLEMENTATION

Since the correspondence problem solution can be very complex, we opted to use C and C++ due to our familiarity with said programming languages. Mathworks MATLAB was used to evaluate some parts of algorithms discussed in one dimension for illustrative purposes. We chose to run all stereo image experiments using C and C++ in the Microsoft Visual Studio design environment. C and C++ can implement the complex procedures of the correspondence problem and offer a reasonable degree of memory control. Many other implementations are feasible but not investigated in this research.

E. RELATED WORK

The correspondence problem has been researched heavily in the past using many different techniques. An overview of the various methods that have been investigated can be found in [10].

Dinkar N. Bhat and Shree K. Nayar [11] developed a distance metric κ between rank permutations that demonstrated robustness in stereo correspondence matching. Their research also investigated the performance of κ compared to other similar metrics using standard traditional pattern matching techniques.

Changming Sun [12] developed an effective two stage dynamic program for window based pattern matching that finds the optimal solution to the correspondence problem with minimal computational complexity.

In other efficient implementation research, traditional matching approaches in [5] and dynamic programming approaches in [13] have been investigated. Traditional and dynamic techniques such as these have also been looked at for FPGA implementation but using different pattern metrics including Hamming distance, sum of squared distances (SSD) and sum of absolute differences (SAD).

F. THESIS ORGANIZATION

This thesis is organized as follows: computer vision concepts are introduced in Chapter II while ordinal measures are defined in Chapter III along with the operations necessary to compute the correlation coefficient κ. An application of the ordinal to the traditional greedy method for solving the correspondence problem is illustrated in Chapter IV. Finally, an improvement to ordinal matching using dynamic programming is introduced in Chapter V. The conclusions and recommendations for further research are presented in Chapter VI.

II. COMPUTER VISION AND STEREO MATCHING

A. PRINCIPLES OF COMPUTER VISION

Computer vision is the discipline, similar to image processing and machine intelligence, which interprets data from sensors such as radar, sonar and cameras so that decisions can be made about the real world [14]. Some sources of computer vision are shown in Figure 3. There are many applicable control and guidance problems that could benefit from a computer capable of identifying or tracking objects within space. For example, an autopilot for a car could be an application of computer vision such that the guidance system would track the lines designating lanes in the road and steer the car to keep it driving safely within a lane. If the system used a video camera as the primary sensory input, this particular problem would require the computer to identify what parts of the video sequence were the lines on the road versus the background and other cars on the road. Typically, the computer would preprocess the video to accentuate a certain feature unique to the object(s) of interest, such as the edges of the lines. The computer could then perform thresholding of the pre-processed data to make a ‘yes’ or ‘no’ decision as to whether the feature belonged to a line on the road. Often, however, this form of segmentation is highly subject to statistical variation (lighting and camera parameters, such as gamma) that can disturb the decision-making ability of the computer. This is because most segmentation algorithms are forced to make assumptions about the video sequence being statistically stationary, which may not always be true. Nevertheless, computer vision algorithms endeavor to distill information from raw sensor data to a level that a computer can understand and base decisions on.

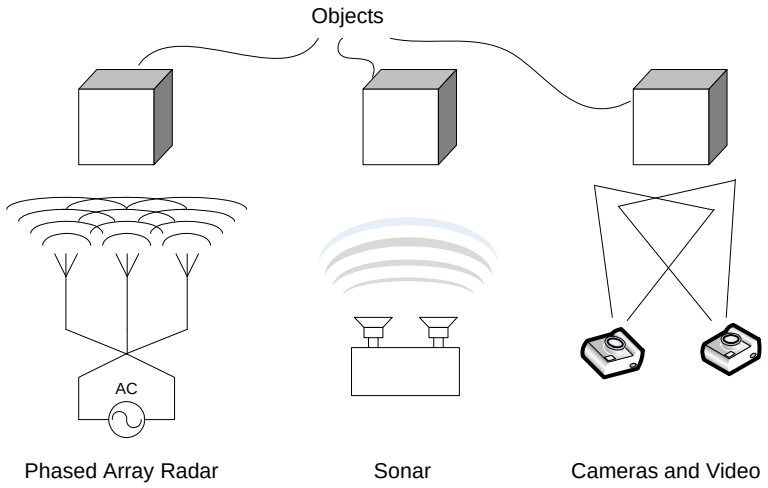


Figure 3. Computer Vision Examples–Radar, Sonar and Cameras.

Decision making for segmentation in computer vision can be taken to a slightly higher level using labels to identify objects of interest in space. In the aforementioned line and autopilot example, a computer could identify a line against non-line components of an image by assigning a '1' to a pixel representing a line and a '0' to a pixel representing anything else. This is called a binary image, since all the pixels of the image are of the binary set $\{0,1\}$. A computer can assign any arbitrary value (not restricted to '1' or '0') to an object, such that unique objects are given a unique "label" value. Fundamentally, the computer will still be restricted to logical 'yes' or 'no' decisions regarding the sovereignty of objects in space, but labeling does give the computer the capacity to distinguish multiple objects from one another. This leads to a broader application of the computer's decision-making capabilities. In the autopilot example, a line could be distinguished from other objects by means of a threshold, which forces all data above the threshold into one category and all data below the threshold into the other category. Perhaps, by means of another algorithm, image information could be distilled into a variety of categories, with each category having its own label. For example, objects can be labeled into categories based on their spatial ordering, increasing order from the top left of the image or video sequence to the bottom right. The computer can make smarter decisions by testing aspects of each category, or type, of element within the

distilled information. Categorical labeling provides a computer more latitude in object segmentation while still only using binary decision making.

As an alternative to the basic edge detection and threshold pre-processing algorithm, algorithms that solve the correspondence problem can translate pixel intensity values of an image into categorical information unique to objects based on their location in space. Traditionally, a computer can segment an object from a video sequence or scene as pixels on or around said object change intensity. For example, motion can be detected as changing intensities of pixels around objects. Only one video sequence is necessary to detect motion of an object. However, given a special video sequence of two separate views of the same scene—called a "stereo image sequence," a computer can potentially compute spatial attributes of objects in the scene and distinguish object uniqueness. This allows the computer to extract spatial properties of objects that can be categorized and labeled.

Stereo images depict a scene or object in focus from two different angles. The focal points of a pair of cameras (such that they form a stereo pair) would ideally converge on the object or area in the space of interest. Since cameras sample the three-dimensional world and translate it into a two-dimensional representation, each camera in a stereo pair distorts the scene in a subtly different way. The distortion between the two cameras' perspectives is inversely proportional to the depth, representing the third axis bisecting the angle formed by the focal lengths of each camera (shown in Figure 4).

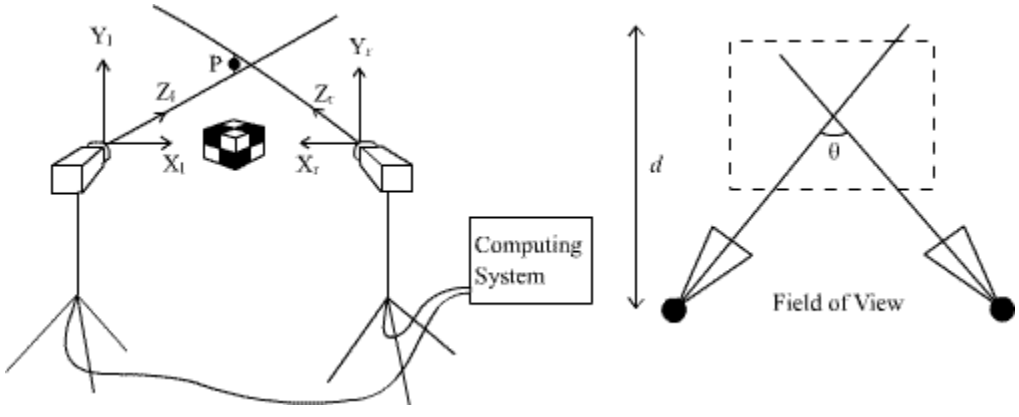


Figure 4. Stereo Pair of Cameras Observing an Object.

Stereo matching algorithms attempt to solve the correspondence problem. The correspondence problem is a challenging computer science problem wherein a set of points from one image are identified in the other image of a stereo pair. The human visual system solves the correspondence problem in real time by matching patterns and edges seen from one eye to patterns and edges seen from the other eye [14]. There are many different algorithms, but some robust algorithms rely on matching patterns within windows around each point of interest. Matches are determined using metrics that compute the similarity of a window from the left image to a window from the right image. The pair of windows with the greatest similarity is deemed a match and the disparity of the points can be computed. Disparity is defined as the translation of a point from its location in one image to its location in the other image and is inversely proportional to depth. Thus, a point that moves considerably between the two images is closer to the stereo pair of cameras than a point that moves only slightly or not at all. If a pixel represented an object located precisely at the intersection of the focal lengths of two cameras, the point's translation would be equal to zero and thus represent both zero disparity and infinity depth. This effect is reduced as the two cameras' focal lengths become parallel; but, as the focal lengths approach parallel, the sensitivity of the system to depth variation is reduced until nearly completely washed out by a very large constant disparity and zero field of depth.

The stereo correspondence problem can be considered in either a general case or a specific case. The general case of the correspondence problem makes no assumption about the scene other than that there are physically points in the scene that correspond. The general case often requires more processing because the search for matches must occur in both dimensions of each image to be successful. The specific case makes use of known parameters about the cameras and the scene they observe such as the distance between the cameras, the focal lengths or the angles between the camera focal lengths [15]. This allows for more precise algorithms that can take advantage of triangulation to accurately compute the true depth of the scene. The specific case of stereo correspondence often employs algorithms that correct the scan lines (the true horizontal lines of the images with respect to their rows that make up the pixels in the image) of one

image with respect to the scan lines of the other [15], [16]. This allows the computer to consider the correspondence problem in only one dimension. For the purposes of this thesis, it is assumed that parametric data about the stereo pair is unknown and thus only the general case of the correspondence problem will be considered.

Stereo matching algorithms solve the correspondence problem to create a disparity map representing all the depth information between the two cameras [10]. The disparity map is essentially distilled information about all the objects in view of the two cameras. A computer can process the disparity map by placing the measured disparity values into categorical bins representing sovereign objects and their locations. This segmentation of objects is thus based on spatial properties rather than intensity based pre-processing typical edge detection and threshold illustrated in the car auto-pilot example. A disparity map of a real scene will predominantly consist of surfaces, seen as regions of constant or constantly changing disparity values. Distortions of disparity information within a region of constant disparity often represent an object in front of or resting on a surface. Ultimately, the disparity map typically offers a computer more information useful for object segmentation than any threshold algorithm, allowing the computer to discriminate objects using a labeling technique intrinsic to the level-like nature of disparity maps. An example is shown in Figure 5 where the disparity map is on the right, and its associated reference image from which it was derived is on the left. The higher intensities in the disparity map equate to larger disparity vectors and, in general, the closer the objects represented by those vectors. The original image on the left hand side is just one of a pair of stereo images (not shown for convenience).

Figure 5. Example Disparity Map.

B. STEREO MATCHING CHALLENGES

Generally speaking, computers do not respond well to solving problems involving far from ideal conditions, a limitation particularly plaguing to the correspondence problem. In the best of circumstances, the correspondence problem requires the computer to measure a distortion, which, in other words, is the computer attempting to match two similar but often non-identical features. This requires the computer's measuring mechanism to have some looseness in determining if features or patterns actually match, avoiding false negative matches, while sustaining some degree of discrimination so as to minimize false positive matching.

While the problem is inherently problematic for a computer, computers and algorithms must contend with the fact that even miniscule variations between the two cameras can severely stunt an algorithm's matching abilities. For example, if one camera has a slightly different focus from the other, it is possible that while the cameras are observing the same scene, there will be no matching patterns or features between the stereo pair of images. This effect will be observed later and is difficult to mitigate in the matching process. Another problem for algorithms solving the correspondence problem using statistical metrics for match determination is that matching could become problematic if images have varying statistical characteristics as shown in Bhat and Nayar [17].

Stereo images themselves introduce distortions as a result of sampling a three-dimensional object or scene as a two-dimensional image. For this thesis, this type of distortion or error will be referred to as occurring *outside the system* and cannot be mitigated but can potentially be interpolated as a result of spatial correlation. There are four types of distortion: depth discontinuity, occlusion, specular reflection and projective distortion:

1. Depth discontinuity commonly occurs when the normal of a surface is very close to perpendicular to one camera's focal length but still visible and completely unobserved by the other camera. Surfaces that create depth discontinuities contribute to the non-linear nature of the disparity map.
2. Occlusion occurs when an object or surface in the foreground covers objects or surfaces in the background so that the background behind the object appears differently to either camera. Thus, no matching points exist in the region of occlusion when an occlusion occurs.
3. Specular reflection causes a pattern to occur in differing locations in each image while not being consistent with the true depth profile of the scene such as a mirror.
4. Projective distortion, the most common form of distortion in stereo matching and partially touched on before, occurs as a result of sampling a three dimensional object or scene as a two-dimensional image.

These distortions are illustrated in Figures 6 through 9.

Figure 6. Example of Depth Discontinuity.

Figure 7. Example of Occlusion.

Figure 8. Example of Specular Reflection.

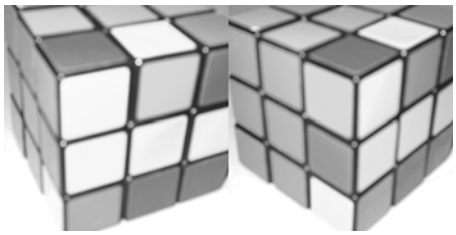


Figure 9. Example of Projective Distortion.

Distortions frequently introduce matching error that will need to be compensated for. Earlier implementations of stereo matching algorithms handle distortion in hindsight as though the information did not exist. More advanced algorithms overcome error by finding the optimal solution (match) even when a good match does not exist [18]. Both implementations will be investigated in detail in later chapters.

C. DISPARITY MAPS AND OBJECT SEGMENTATION

The disparity map is inversely proportional to the depth observed by the stereo pair of cameras. A disparity map will be referred to in this thesis as a *solution* and is commonly very dense, having a *solution* for the translation of every point in an image to every corresponding point in the other stereo image. Disparity maps are made up of vectors in Cartesian coordinates but are rendered as an intensity image such that the magnitude of the vectors at each point is proportional to the intensity. For example, if an arbitrary point P in *Image 1* were to move three pixels right and one pixel down to P' in *Image 2*, the intensity of the disparity map at that point would be proportional to the magnitude 3.16 as demonstrated in Figure 10.

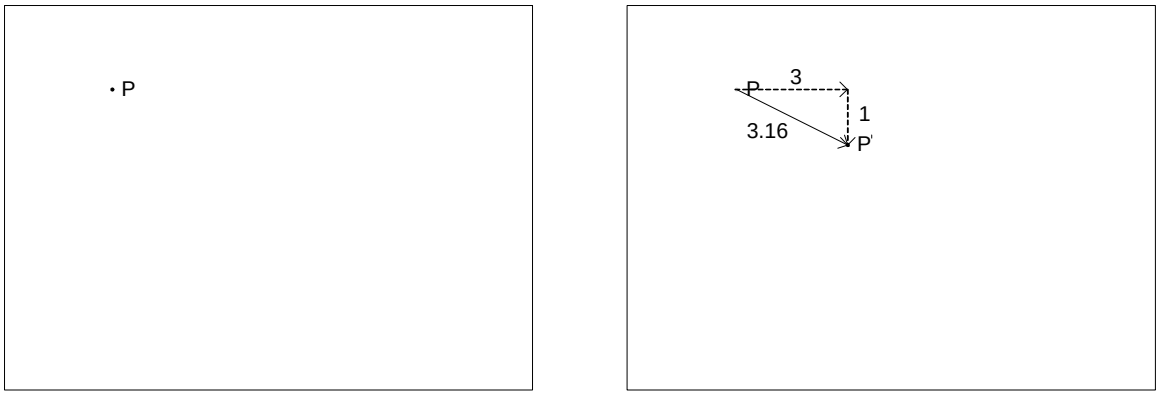


Figure 10. Example of a Disparity Vector.

Because natural images generally have high spatial correlation in two dimensions, this generalization applies to the third dimension measured by stereo correspondence algorithms. As a result, objects commonly have similar or consistent disparity vectors. Computers can take advantage of this spatial correlation to segment objects based on the magnitude of the disparity vectors as depicted in Figure 11.

Figure 11. Example Stereo Pair and Disparity Map with Distinctive Object. After [19], [20]

Notice that the object in the foreground can be distinguished from the background because it is represented by a region of consistent disparity vectors. Unlike edge detection based methods that use intensity values of pixels of an image, the object

is *spatially* segmented from its surrounding using the disparity map because the information in the disparity map is directly related to depth.

Another feature of disparity maps is that the precision of a disparity map increases as the resolution of the reference images (the stereo pair) increases. Higher resolution images can have objects consisting of more pixels and patterns. Thus, more pixels means that while the location of an arbitrary point is constant in space, the number of pixels separating points P and P' representing the point increases with the resolution of the stereo pair as shown in Figure 12. Subsequently, the disparity vectors become proportionally more precise.

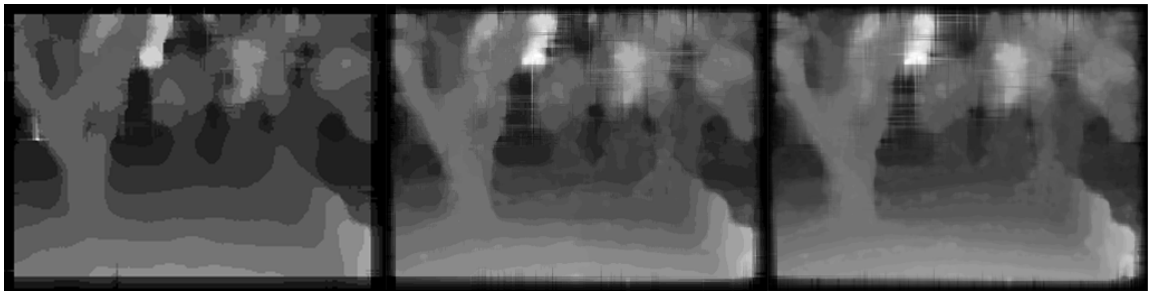


Figure 12. Comparison of Low and High Resolution Stereo Images.

In the next chapter, the ordinal coefficient κ is introduced. The ordinal is a logical metric that has advantages for hardware implementation. A proposed architecture, also introduced in the next chapter, computes the correlation coefficient κ in parallel and fully pipelined on an FPGA.

III. ORDINAL MEASURES AND THE CORRELATION COEFFICIENT *K*

A. WINDOW COMPARISON METHODS

Area based matching algorithms use windows to match corresponding points between two stereo images. Ideally, in a stereo pair with infinite spatial resolution, a pixel alone would be uniquely identifiable and, thus, can be matched from one image to another. However, since real pixels have finite spatial resolution, they do not carry enough information to be matched without additional pattern-based information of the neighborhood surrounding a pixel. Thus, point-by-point matching for stereo correspondence in the time domain becomes pattern matching with window based methods, matching patterns around points shown in Figure 13.

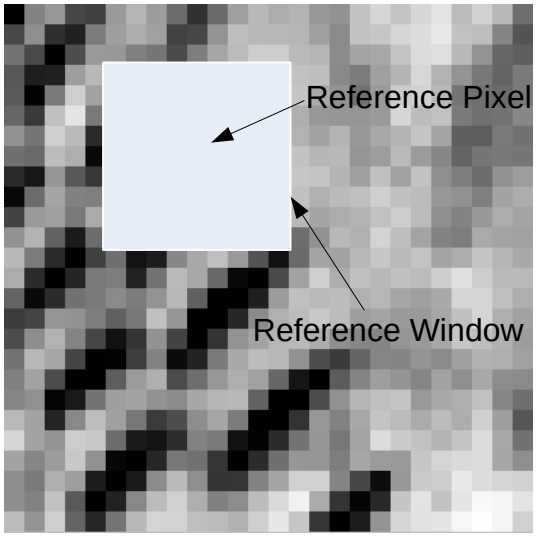


Figure 13. Defining a Pattern using a Window around a Point in the Reference Image. After [19]

Window based methods generally compute a coefficient proportional to the similarity of patterns between two windows [10]. This coefficient is called a correlation coefficient. Thus, finding a "best match" is an optimization problem where a match is determined by maximizing (or minimizing) a matching method's correlation coefficient. Often in images

with very high signal-to-noise ratios (SNR), simple optimization algorithms are enough to determine matches. Also, window based methods generally rely on a standard pattern matching technique. First, a reference point is chosen sequentially from the reference image (perhaps the left image for demonstrative purposes). A window is then built around this point so that a reference pattern is used to represent that point. Next, the target point(s) are chosen within a search range of the target image (the right image in this example). Windows are also constructed around these target points (overlapping allowed) to form representative patterns around each target point. Finally, the window based correlation computation is applied to all target patterns (windows), comparing each target pattern with the reference pattern. The process is illustrated in Figure 14. A match is determined by maximizing the correlation coefficient among all possible matches. The pattern based window matching is analogous to matching the individual points or pixels of the two images provided that the windowing scheme is consistent.

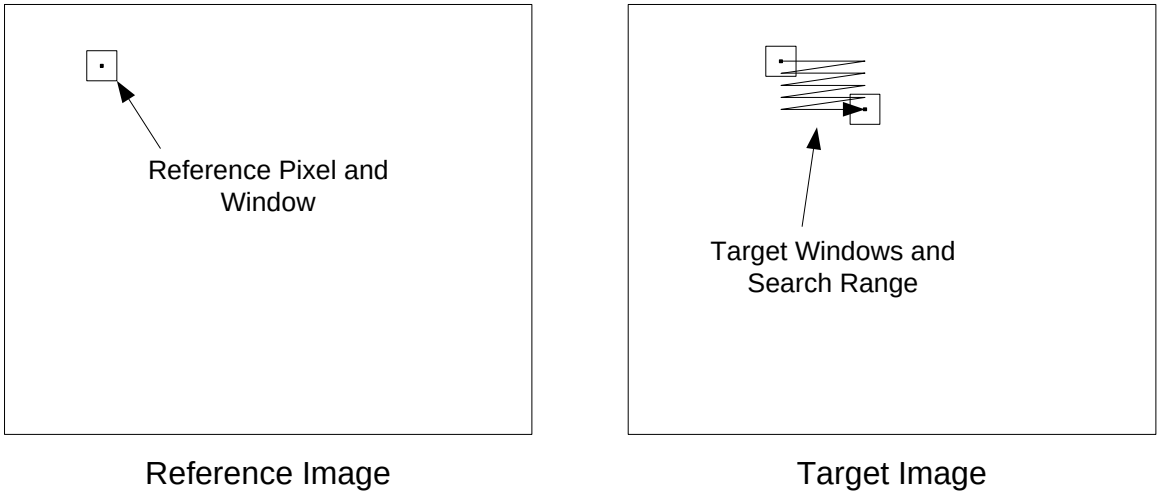


Figure 14. Pattern Matching Technique.

Window matching methods use search ranges to identify the best match among a candidate region of possible matches. Often, search ranges account for the majority of computational complexity in any pattern matching technique. Search ranges manifest as deeply nested loops or loops within loops. For example, for every reference point in one image, the algorithm will need to find its best match within a search range of five rows

and fifteen columns. This equates to 75 simple match calculations per pixel in the reference image, which can be a staggering increase in computational time even for small images. Often, simple matching calculations are not enough and more complex matching techniques are needed and will be explored in depth later. Search ranges also intensify the computational issues of the chosen correlation coefficient.

The correlation coefficient computed for a window appraises the similarity of the patterns within the pair of windows. Many of these measures use statistical means to compute the correlation coefficient such as the sum of absolute differences (SSD) given by

$$SSD = \sum_{i=1}^{n} \left(I_1^i - I_2^i \right)^2 \tag{1}$$

or the normalized correlation coefficient (NCC) given by

$$NCC = \frac{\sum_{i=1}^{n} \left(I_1^i - \bar{I}_1 \right) \left(I_2^i - \bar{I}_2 \right)}{\sqrt{\sum_{i=1}^{n} \left(I_1^i - \bar{I}_1 \right)^2} \sqrt{\sum_{i=1}^{n} \left(I_2^i - \bar{I}_2 \right)^2}} . \tag{2}$$

Notice that the computations for Equation (2) are not trivial. Other metrics use filtering techniques to compute the similarity to a reference window at each point in the target image. These methods are subject to outliers in pixel intensities such as an image pair corrupted with salt and pepper noise. Other measures such as the ordinal are dependent on the ranking of pixels and their relative ranking to each other. Ordinal metrics are very robust to statistical variations between the two images such as gamma and contrast issues but can be frustrated by additive white Gaussian noise (AWGN) as shown in Bhat and Nayar [11]. Statistical measures can be very computationally expensive, requiring many multiply and add operations per window comparison. The measure of interest for the research conducted is the ordinal metric that uses the rank permutations of pixels in windows to compute the correlation coefficient κ.

B. ORDINAL MEASURES WITH *K*

Ordinal measures are used to compute and compare the rank permutations of candidate windows to compute the correlation coefficient κ. The ordinal κ has been shown to be very robust in comparison to statistical correlation metrics such as SSD and NCC [11]. The ordinal also has computational advantages for hardware implementation and robustness in choosing matches that are investigated in later sections.

The goal is to define a measure of correlation between the two sequences which is least sensitive to random variations of individual values and, at the same time, easy to compute on sequential and parallel machines.

This leads to an approach based on the ordering of the values rather than the values of the sequences themselves. For example, take three sequences of length three as $x_1 = [2.5, -3.2, 1.7]$, $x_2 = [1.9, -5.1, 0.8]$, and $x_3 = [0.1, 2.5, 0.9]$. One can see that, by the ordering of the values, x_1 and x_2 are similar, while x_1 and x_3 are not since, in this case $x_1(2) < x_1(3) < x_1(1)$, $x_2(2) < x_2(3) < x_2(1)$, and $x_3(1) < x_3(3) < x_3(2)$. It can be seen that the sequences x_1 and x_2 have the same "ordering" of values while x_3 is different.

In this section, the basic theoretical framework of definitions and metrics is introduced so that the similarities of sequence ordering can be assessed. In particular, the following is defined:

Definition 1: For any integer n call S_n the set of all permutations of the indices $1,2,...n$. For example, $S_3 = \{(1,2,3),(1,3,2),(2,1,3),(2,3,1),(3,1,2),(3,2,1)\}$. Clearly, the "cardinality" of S_n (i.e., the number of elements in the set) defined as $|S_n|$ is computed as:

$$| S_n | = n! \tag{3}$$

Definition 2: Given a set of index permutations S_n, call $\pi = (\pi^1, \pi^2, ..., \pi^n) \in S_n$ an element of the set. For example, (again for $n = 3$) choosing $\pi = (3,1,2)$ corresponds to

$\pi^1 = 3, \pi^2 = 1, \pi^3 = 2.$ By the very nature of these definitions, each permutation $\pi \in S_n$ can be seen as a mapping between the indices $(1,2,..,n)$ and the ordering is

$$\pi : (1,2,...,n) \to (\pi^1, \pi^2, ..., \pi^n). \tag{4}$$

Any sequence x of real numbers can be converted to its associated permutation π by

$$\pi^i = \sum_{j=1}^{n} J\left(x(i) > x(j)\right) + \sum_{j=1}^{i} J\left(x(i) = x(j)\right), \tag{5}$$

where $J(B) = 1$ if the expression B is true, $J(B) = 0$ if the expression B is false, and i is the specific sample in the vector x being evaluated.

An example of transforming two data sets x_1 and x_2 into each respective rank permutation π_1 and π_2 is shown in Figure 15.

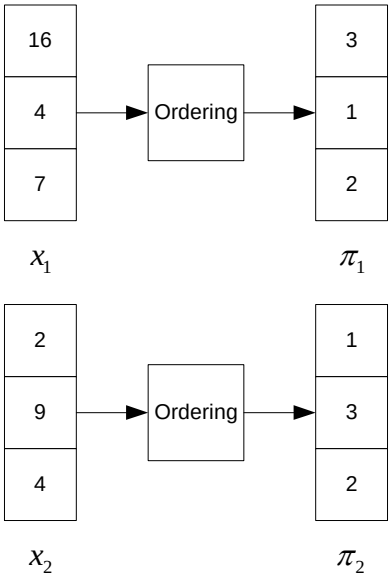


Figure 15. Example Rank Permutation Conversion.

This leads to the definition of inverse permutation as follows:

Definition 3: Given any permutation $\pi \in S_n$, the inverse permutation $\pi^{-1} \in S_n$ is defined as the mapping

$$\pi^{-1} : (\pi^1, \pi^2, ..., \pi^n) \to (1, 2, ..., n) \tag{6}$$

where $\pi^i = k$ and $\left(\pi^{-1}\right)^k = i$ for $i, k = 1, ..., n.$

For example, again in S_3, the permutation $\pi = (3,1,2)$ can be seen as the mapping $\pi : (1, 2, 3) \to (3, 1, 2)$. Its inverse is the mapping $\pi^{-1} : (3,1,2) \to (1,2,3)$ which, after reordering, becomes $\pi^{-1} : (1, 2, 3) \to (2, 3, 1)$.

This leads to the definition of the positive and negative identity permutations $I_+ = (1, 2, \ldots, n)$ and $I_- = (n, n-1, \ldots, 2, 1)$. The identity permutations have the property that the inverse of each identity permutation is itself such that $I_+^{-1} = I_+$ and $I_-^{-1} = I_-$. An example of the processing of the inverse permutation is shown in Figure 16.

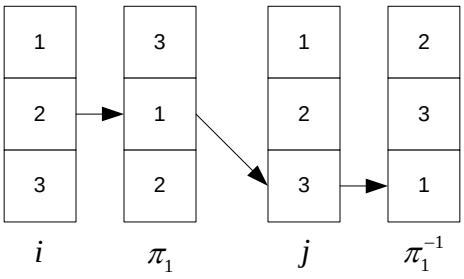


Figure 16. Example Inverse Permutation π_1^{-1}.

Definition 4: Any two permutations can be combined into a single permutation s such that $s \in S_n$ where s defines a permutation π_2 mapped to π_1^{-1} and re-ordering the indices of s as $1, 2, \ldots, n.$ The formulation is given by

$$s : \left(\left(\pi_1^{-1}\right)^1, \left(\pi_1^{-1}\right)^2, \ldots, \left(\pi_1^{-1}\right)^n\right) \to \left(\pi_2^1, \pi_2^2, \ldots, \pi_2^n\right) \tag{7}$$

where $s^i = \pi_2^k$ and $k = \left(\pi_1^{-1}\right)^i$.

For example, given $\pi_1^{-1}=(2,3,1)$ and $\pi_2=(1,3,2)$, then $s:(2,3,1)\to(1,3,2)$ and re-ordering the indices of s as $1,2,\ldots,n$, we get $s=(2,1,3)$. An example of computing the combined permutation s from π_1^{-1} and π_2 is shown in Figure 17.

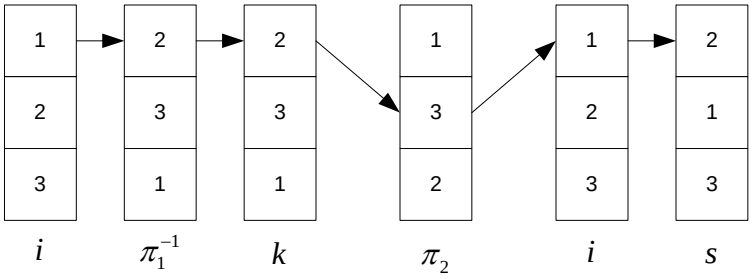


Figure 17. Example Combined Permutation s.

Definition 5: A distance vector d_m is the relationship between s and the identity permutation I_+. When $s=I_+$, the distance is zero such that $d_m=(0,0,\ldots,0)$, indicating perfect positive correlation, and when $s=I_-$, d_m achieves a maximum distance of $\left\lfloor \frac{n}{2} \right\rfloor$, indicating perfect negative correlation. Finally, d_m^i is defined by

$$d_m^i = i - \sum_{j=1}^{i} J\left(s^j \le i\right) = \sum_{j=1}^{i} J\left(s^j > i\right) \tag{8}$$

where $J(B)=1$ if the expression B is true and $J(B)=0$ if the expression B is false. For example, given $s=(2,1,3)$, then $d_m=(1,0,0)$.

Definition 6: A normalized correlation coefficient κ is the measure of correlation between π_1 and π_2 taken as the maximum distance in the d_m vector. If the input random sequences are perfectly correlated, κ takes on the normalized value 1. If the input random sequences are perfectly un-correlated, κ takes on the normalized value -1. The coefficient κ is defined as

$$\kappa(\pi_1,\pi_2) = 1 - \frac{2\max_{i=1}^{n} d_m^i}{\lfloor n/2 \rfloor}, \tag{9}$$

where π_1 and π_2 are permutations to be compared. Finally, the overall flow chart of data from sample data windows x_1 and x_2 to computing the correlation coefficient κ is shown in Figure 18.

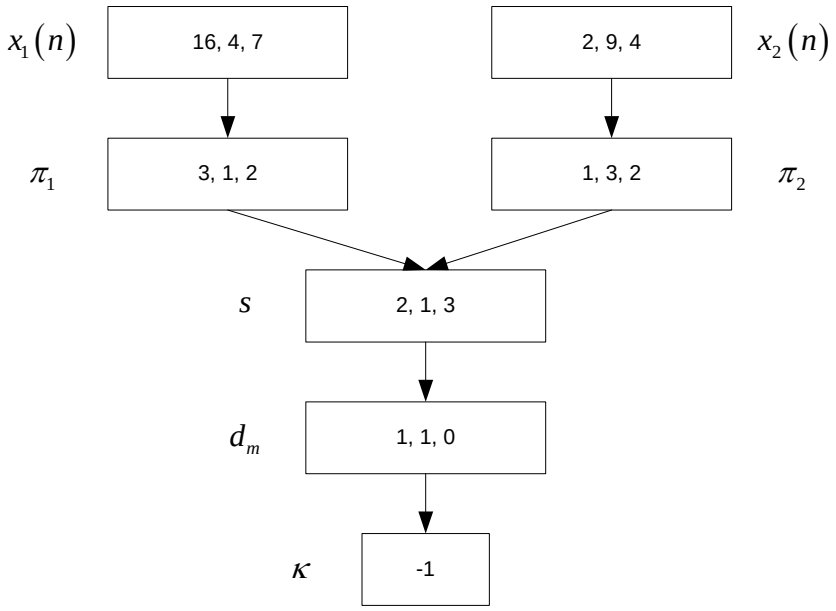


Figure 18. Ordinal Processing Flow Chart. After [11]

C. HARDWARE IMPLEMENTATION ADVANTAGES

Computation of the correlation coefficient κ has significant advantages for hardware implementation. First, the coefficient κ can be computed in a purely logical fashion requiring no summations or multiplication. Thus, in a hardware implementation, only basic logic slices of comparators and multiplexers are needed to realize the ordinal measure. For illustrative purposes, the hardware implementation will be explored.

1. Rank Permutations

The first component needed to compute the correlation coefficient κ is to compute the rank permutations of each window. In software algorithms, rank permutations can be calculated quite quickly, especially when using the counting sort algorithm. Computation of rank permutation is inefficient with $\Theta\left(n^2\right)$ complexity. With

hardware, the naïve sort's worst case scenario complexity is taken and spread out spatially to compute the rank permutation in full parallel to achieve computational complexity $\Theta(1)$ in time, or one clock cycle (given a typical sized input of n 8-bit pixels to be sorted), but $\Theta(n^2)$ in space. For stability in the rank permutation, the hardware requires an additional $\Theta(n^2)$ instantiated complexity but with no additional computational time. The architecture for computing the rank permutation of a two pixel window is shown in Figure 19.

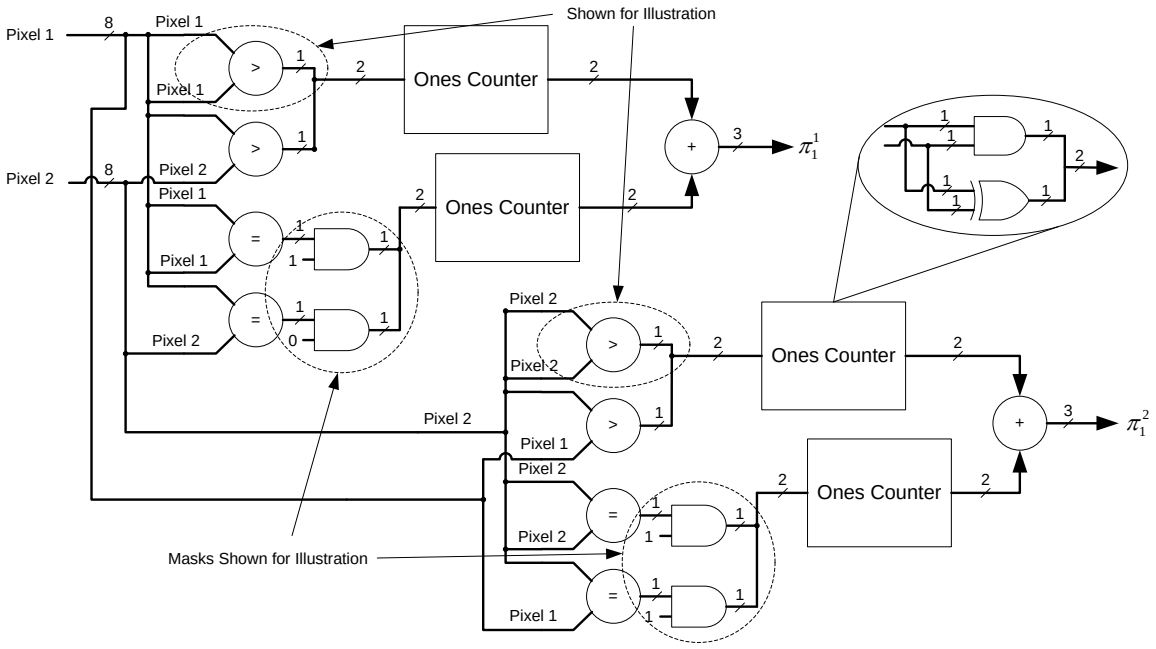


Figure 19. Rank Permutation Architecture.

The rank permutation circuit shown in Figure 19 needs eight comparators to compute the ranking of two pixels. The input to the circuit is each 8-bit pixel intensity in the window to be ranked. Each comparator outputs a logic '1' if the inputs satisfy the Boolean expression in the comparator. The ranking of a pixel at π_1^i is the summation of the number of pixels in the window less than the i^{th} pixel plus the number of pixels equal to the i^{th} pixel. The summation is carried out by a logical "ones counter" circuit rather than actual adders. For very large bit vectors (over 128 bits), the ones counter circuit can

be realized as a sum-of-products or product-of-sums combinational circuit to minimize delay. Also, the equal comparators are masked to only be capable of summing up to i. This ensures stability in the rank permutation by assigning rankings of equal intensity by the order of the pixels in the window. The output of the rank permutation is a vector of ranks encoded in binary coded decimal.

Computation of correlation coefficient κ requires two rank permutation components. The rank permutation hardware makes up the majority of the metric's hardware implementation logic requirements. Because of the $2n^2$ instantiated complexity of the architecture, larger windows of nine by nine or 11 by 11 require significant logic to fully implement. With the proposed architecture, computation can typically be handled in one clock cycle for nine by nine or 11 by 11 sized windows of pixels.

2. Inverse Permutation and the Combined Permutation s

The inverse permutation is tricky to compute in hardware. The simplest implementation of the inverse permutation is to define two registers, one n sized register and one n^2 sized register. For each rank, a set of n selectors compare the ranking at π_1^i with a set index i. If the ranking is equal to the index, the rank is multiplexed to the new location by π_1^i and all other rankings in the target portion of the register are set to zero. The first register holds the rank permutation, and the second register holds an expanded form of the inverse permutation. The expanded inverse permutation is to ensure that flip-flops in the register are never driven by more than one multiplexer. The expanded inverse permutation is condensed by logical or from n^2 to n. A proposed architecture for computing the inverse permutation for a window of two pixels is shown in Figure 20.

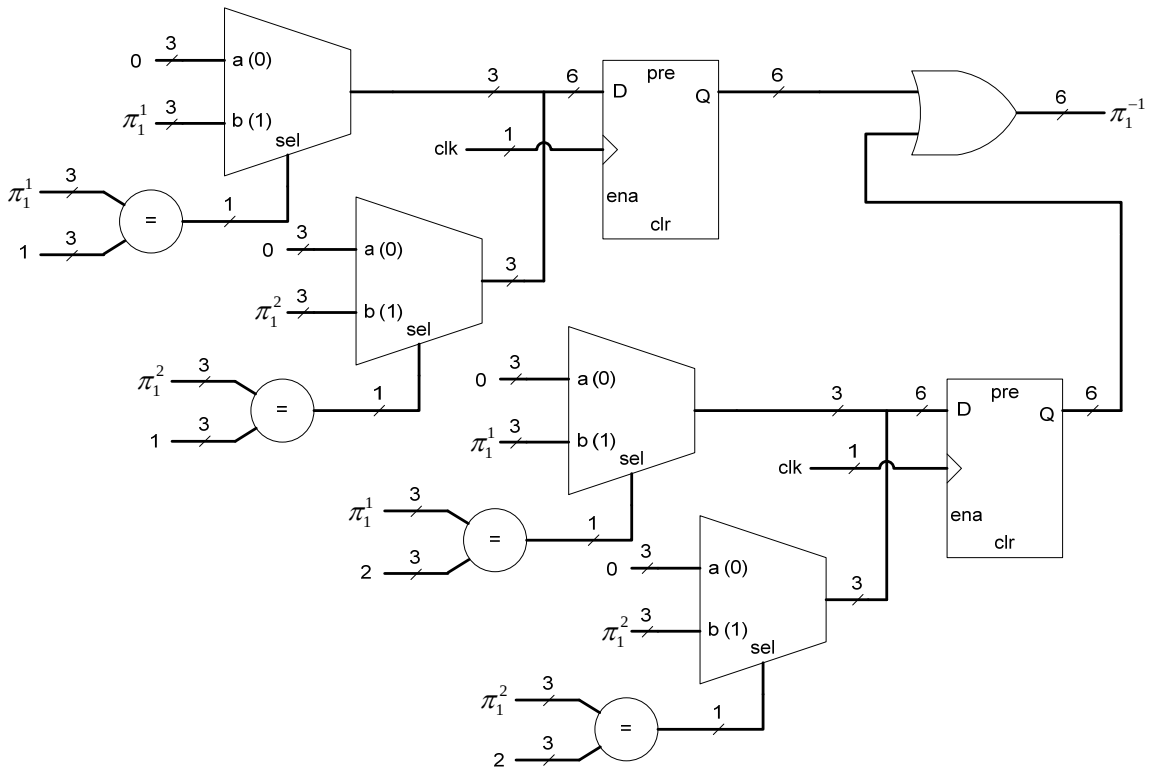


Figure 20. Inverse Permutation Architecture.

Next, the combined permutation is easily calculated by multiplexing the non-inverted rank permutation with the inverted permutation as the control. The proposed combined permutation architecture is shown in Figure 21. Note that the inverse permutation elements $\left(\pi_1^{-1}\right)^1$ and $\left(\pi_1^{-1}\right)^2$ can each only take on values $[1,2]$ unless a bit error has occurred. The inverse permutation is directly ported into n selectors that are n wide, passing the associated rank to the output with respect to the i^{th} index of the inverse permutation.

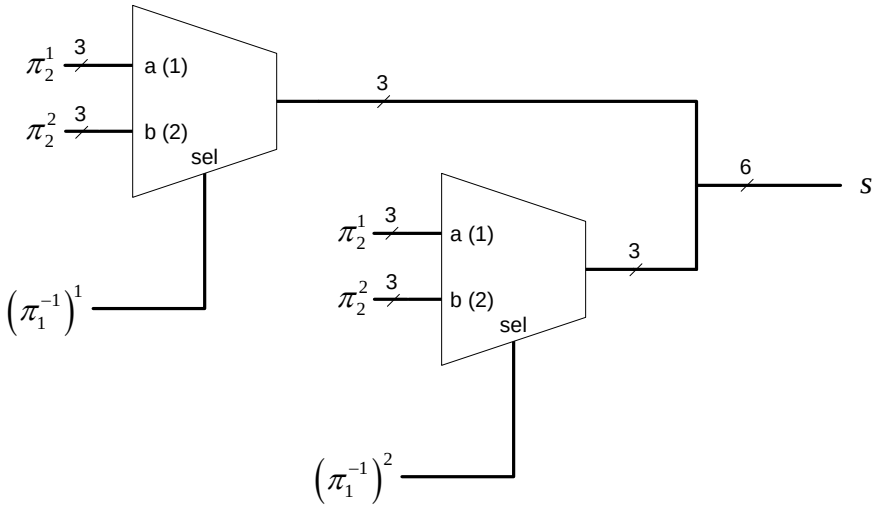


Figure 21. Combined Permutation Architecture.

3. Distance and Correlation Coefficient κ

The most complex part of calculating the correlation coefficient κ with hardware is computing the distance vector. For this, a property of the distance calculation is exploited to parallelize the computation. The distance from the identity permutation is graphically illustrated in Figure 22.

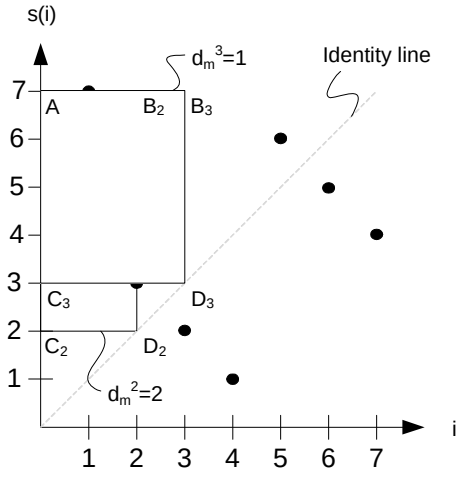


Figure 22. Distance Scatter Plot.

The distance vector between the combined permutation s and the identity permutation is composed of the number of ranks bounded by the region defined by $\left(x \geq 0, x \leq i, y > i, y \leq n\right)$ or the region defined by ABCD in Figure 22 (ranks falling on the bottom boundary are not counted). Since the s permutation is compared to the identity permutation I_+, the inverse permutation s^{-1} represents the ranks out of place with respect to I_+ but *not* bounded by the region. Thus, the number of ranks in s that occupy this region is equal to the number of ranks out of place up to index i minus the number of s^{-1} ranks out of place up to index i. Thus, the distance vector is composed of two separate vectors which are defined as

$$J_{pos}^{i} = s^{i} > I^{i} \tag{10}$$

and

$$J_{neg}^{i} = \left(s^{-1}\right)^{i} < I^{i}. \tag{11}$$

Then, the distance d^i at any index i is defined as

$$d^{i} = \sum_{k=0}^{k \leq i} J_{pos}^{k} - \sum_{k=0}^{k \leq i} J_{neg}^{k}. \tag{12}$$

Equations (10) and (11) are graphically illustrated in Figure 23 using two scatter plots of the s permutation and the s^{-1} permutation. The J vectors can be joined in parallel using Equation (12). The parallel merging of the J vectors is graphically illustrated in Figure 24.

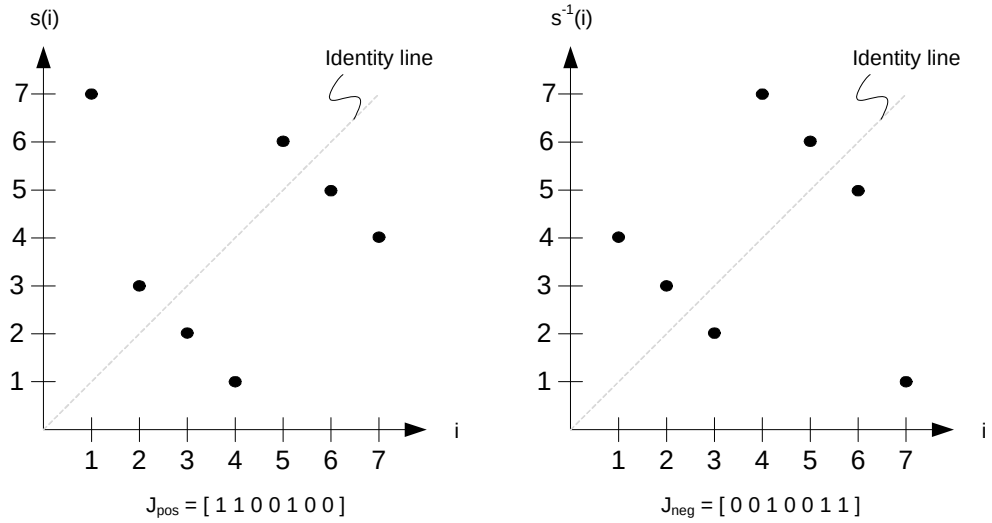


Figure 23. Computing the J Vectors.

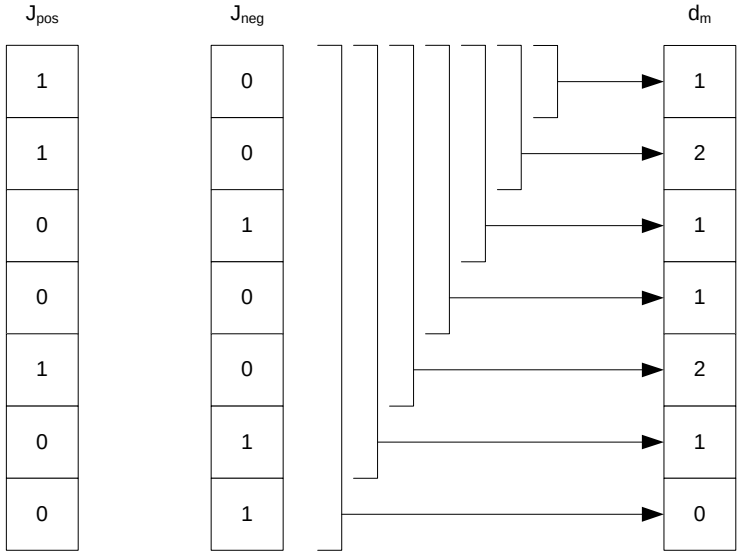


Figure 24. Calculating the Distance Vector in Parallel.

In hardware, the summations of the two J vectors is trivial since any element in either vector is $\{0,1\}$. Thus, the summation is logically a ones counter, which can be implemented without the use of adders or a clock. A proposed hardware implementation is shown in Figure 25.

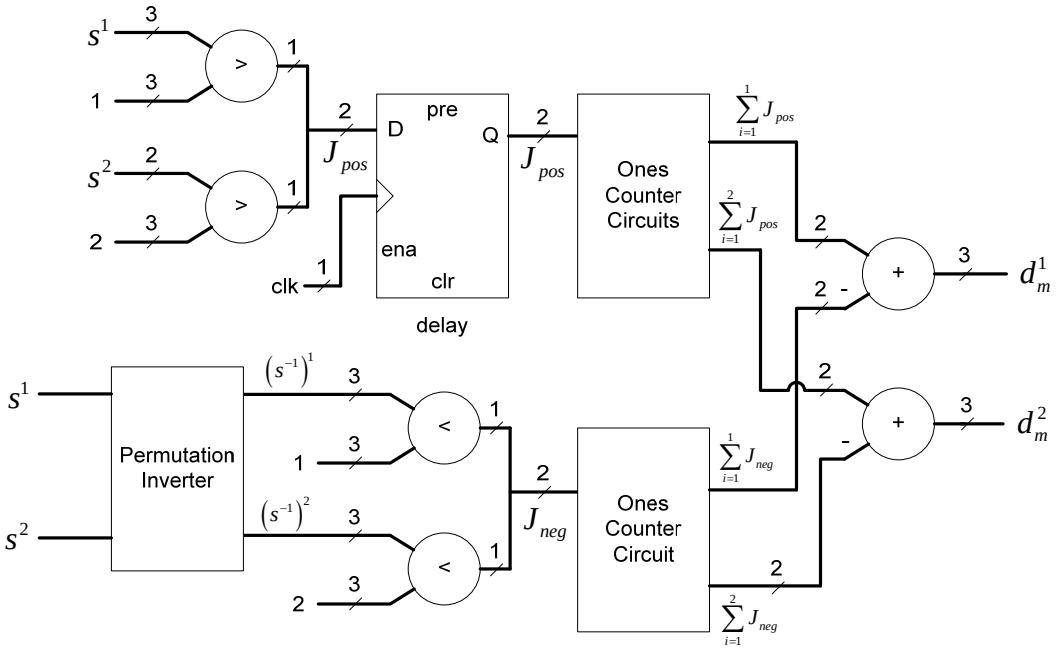


Figure 25. Distance Hardware.

Finally, κ is computed by taking the maximum of the distance vector using a maximums array circuit that logically selects the greater d_m of a pair iteratively by stages until the maximum is found. The maximums circuit requires $\lceil \log_2 n \rceil$ stages to compute where each stage computes a partial maximum with each clock cycle. The κ coefficient need not be normalized to fall between [−1,1] since this would require floating point computations. Instead, the raw maximum distance is used, and the optimization of κ occurs when the maximum distance is *minimized* rather than maximizing the normalized coefficient.

The proposed implementation suggests one clock cycle for both rank permutations, two clock cycles for the combined permutation, two clock cycles for the distance vector and $\lceil \log_2 n \rceil$ clock cycles to compute the maximum distance for a total latency of $5+\lceil \log_2 n \rceil$ at a clock frequency of 100 MHz (clock period of 10 nanoseconds) in an Altera Cyclone family FPGA. The proposed architecture is also fully pipelined and can compute a result every clock cycle when the pipeline is full. The proposed architecture for a nine by nine ordinal ($n = 81$ and operating on 8-bit pixels) is

synthesized using the Altera Quartus II 8.1 software for several Altera devices and the logic utilization is tabulated in Table 1 to illustrate the metric's logic requirements. The ordinal of size nine by nine could feasibly be implemented in an Altera Cyclone II FPGA, an economical FPGA, while reserving the FPGA's multipliers and most register resources for other purposes such as image re-sampling to form Gaussian pyramids which are explained later.

Table 1. Logic Utilization of Ordinal κ in Popular Altera Devices.

Device	Combinational ALUTs	Registers	Multipliers	DSPs
Cyclone II (EP2C35F672C6)	12,065 / 35,000 (~34%)	442	0 / 70 (0%)	N/A
Cyclone III (EP3C120F780C7)	12,065 / 120,000 (~10%)	442	0 / 576 (0%)	N/A
Stratix III (EP3SL150F1152C3ES)	7,986 / 113,600 (7%)	480 / 113,600 (< 1 %)	N/A	0 / 384 (0%)

Now that the ordinal measure has been introduced, in the next chapter the correlation coefficient κ is applied and a basic matching algorithm that uses a traditional greedy method to solve the correspondence problem is investigated.

IV. ORDINAL MATCHING WITH TRADITIONAL MATCHING STRATEGY

Solving the correspondence problem has been studied extensively in computer science for the last four decades [10]. In traditional matching strategy, stereo matching algorithms applied a "greedy" method for choosing correlated windows. A reference window is chosen from the reference image and passed over a region in the target image, computing the correlation values of the reference window to a set of candidate windows. While variations exist that impose certain constraints, the fundamental technique is to choose the match in the target image region with the maximum correlation value. At the time of the ordinal's development, a traditional greedy method prevailed among window-based approaches to solving the correspondence problem.

A. TRADITIONAL MATCHING STRATEGY

Traditional matching strategies use several techniques to enhance the greedy method. Many of these techniques apply constraints to the matching process to reduce the chance of a false match. Some research has applied probabilistic methods to decide the likelihood of a true match, while other approaches have applied Kalman filtering to help reduce false matches [21], [22]. However, these techniques make assumptions about the depth information between the stereo pair as a linear process that can be modeled. The depth process behaves somewhat linearly in image regions with consistent intensity values but can behave very non-linearly along edges and sharp intensity changes common of depth discontinuities. For example, Kalman filtering naturally results in blurring of the depth wherever a depth discontinuity exists. Thus, stereo correspondence algorithms can are hampered by techniques that attempt to improve matching under certain conditions but do not adequately consider the depth process as a whole. For this research, strategies that make as few assumptions as possible about the depth process are considered, and their computational complexity investigated.

1. Multirate Matching With a Gaussian Pyramid

Some traditional matching techniques are ubiquitous among nearly all stereo correspondence algorithms. Multirate processing, in the sense of processing signals at different resolutions, has been shown to both greatly improve matching while also reducing computational complexity [10]. Multirate entails low-pass filtering the stereo pair and down sampling the pair by a integral rate a certain number of times to form a pyramid structure of coarse to fine resolutions. In window based methods, multirate matching offers more image intensity information at each coarser level while maintaining a constant window size at reduced computational complexity. Matching using a coarse-to-fine scheme gives matching algorithms a "first-look" of the depth process without varying the size of windows. The matching process then proceeds to finer resolutions (larger dimension images) using the solution of the previous coarse resolution as a constraint. Multirate takes advantage of the spatial correlation between the different resolutions' solutions and rules out matches in subsequently finer resolutions that are determined to be too far in disagreement from the coarse resolution solution. A Gaussian pyramid is illustrated in Figure 26.

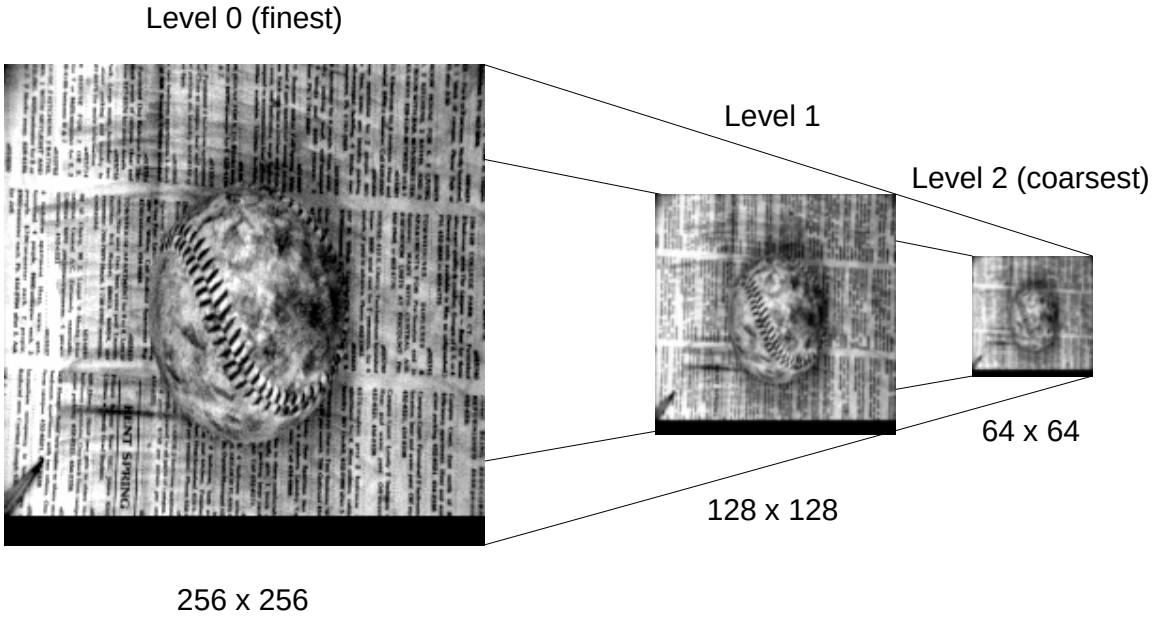


Figure 26. Typical Gaussian Pyramid. After [19]

Before down-sampling an image, it is necessary to low-pass filter the image to reduce aliasing effects. Aliasing can introduce artifacts that distort the matching process by attempting to match the aliased artifacts rather than the valid image information. Also, the aliasing from down-sampling one image in a stereo pair can be substantially different from the aliasing and information in the other image of the stereo pair. A common method in stereo to reduce this aliasing is to filter both images with a two-dimensional Gaussian pulse type low-pass filter. This pulse is defined by

$$K\left(n_1, n_2\right) = e^{-\left(\frac{\left(\frac{n_1 - N}{2}\right)^2 + \left(\frac{n_2 - N}{2}\right)^2}{4\sigma^2}\right)} \tag{13}$$

The standard deviation is arbitrarily chosen to reduce artifacts. For this research, a nine by nine pulse kernel $K\left(n_1, n_2\right)$ (filter window where N is the dimension of the window) with a standard deviation of 0.7 was used with a down-sampling rate of two such that each coarser image is successively half the width and half the height of its finer counterpart. The pulse and frequency response is shown in Figure 27.

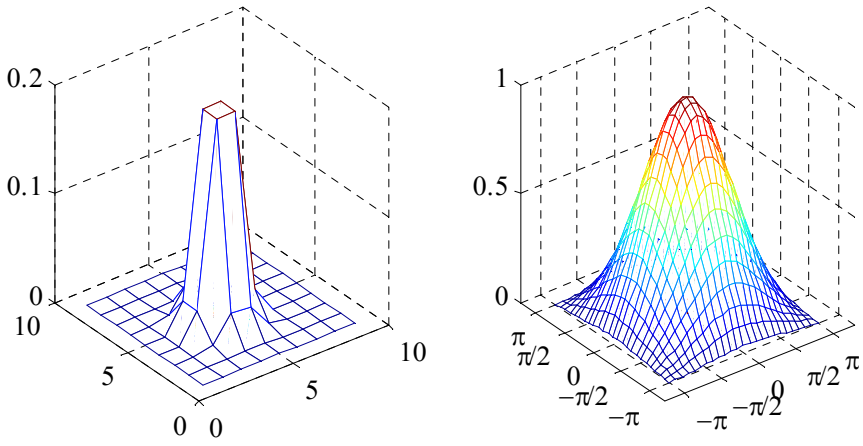


Figure 27. A Gaussian Pulse (9x9) and its Frequency Response.

The pulse is convolved over the whole image, zero padding the image edges to reduce edge effects. After filtering the image, pixels are extracted at the specified rate in

the horizontal and vertical directions to form a lower resolution image. The two-dimensional convolution operation is defined by

$$y(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} I(k_1, k_2) K(n_1 - k_1, n_2 - k_2). \quad (14)$$

The matching process proceeds from the coarsest to the finest image. At the conclusion of the solution of the correspondence problem at a particular resolution, the solution is multiplied by two and up-sampled to the next highest resolution using a zero-order hold. The choice of matches at the finer resolution is confined to the neighborhood of the coarse resolution solution. The initial solution prior to the coarsest resolution is assumed to be zero at every point in the reference image. Each finer resolution solution is a refinement of the previous coarse resolution solution. Thus, getting the correct matches at the coarsest resolution is of paramount importance. A matching error (false match) at a coarse resolution often results in the error propagating to all subsequent finer resolutions. Loosening restrictions on neighborhood matching helps reduce the impact of an error at a coarse resolution.

2. Back Matching Strategy

The greedy method can be constrained by enforcing the matches in the forward direction to agree with matches in the reverse direction (a strategy used by Bhat and Nayar [11]). For example, a target match chosen in the right image by the greedy method must also match by the greedy method within a certain tolerance in the left image when the right image is taken as a reference. This process is illustrated in one dimension in Figure 28, which shows two candidate matches to window 3 in the reference vector: window 4 and window 6 in the target vector. Window 6 is chosen since it matches back to a closer neighborhood.

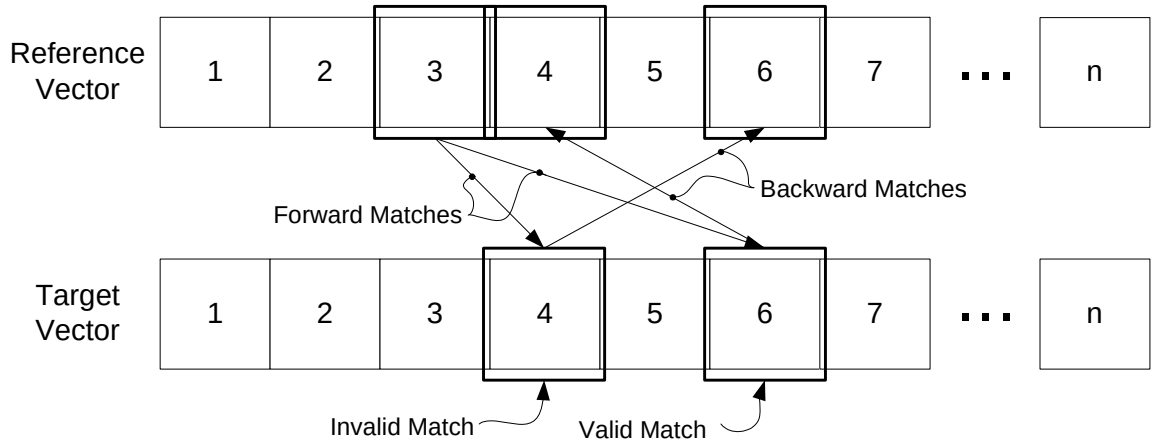


Figure 28. Illustration of the Back Matching Strategy.

The back matching strategy helps eliminate candidate matches that do not agree in the reverse direction. This commonly occurs in occlusion regions where an occluded pattern may not exist between the two images. Therefore, the back matching strategy aids in preventing occluded patterns from choosing false matches. Back matching generally improves the probability of choosing a true match under normal conditions while not making any assumption about the underlying depth process.

The back matching strategy is the core of the traditional algorithm approach to the correspondence problem used in this research. Given a random, one dimension signal x_1 as an input to the algorithm and a shifted version of x_1 corrupted by noise as the other input signal x_2, the associated calculated disparity d and correlation coefficient κ for every matched pattern are shown as the output of the algorithm. Figure 29 shows a basic block diagram of the inputs and outputs of the algorithm.

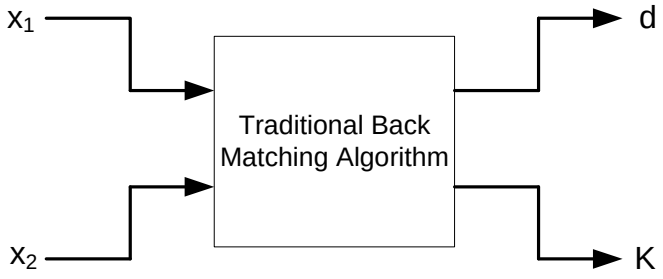


Figure 29. Block Diagram of Basic Back-Matching Strategy Algorithm.

Each disparity d is calculated by rectangular windowing nine samples from the reference signal x_1 and computing the κ across the entire target signal x_2. The top three matches by the correlation coefficient κ are recorded. Each top match window is taken as a reference and back matched by computing the correlation coefficient κ. If the back matching calculates the top match in the reverse direction to be within one pixel of the original reference window with respect to each window's first sample, then a disparity d is calculated by taking the index of the first sample in the matching window in signal x_2 minus the index of the first sample in the matching window in signal x_1. The associated correlation κ for the match pair and disparity d are recorded with respect to the first sample of the window in signal x_1. The result is shown in Figure 30. It is clear from the example that errors still occur using the back matching strategy since the true disparity is $+2$ for all points. The last eight samples are zero due to edge effects with the size $w = 9$ window.

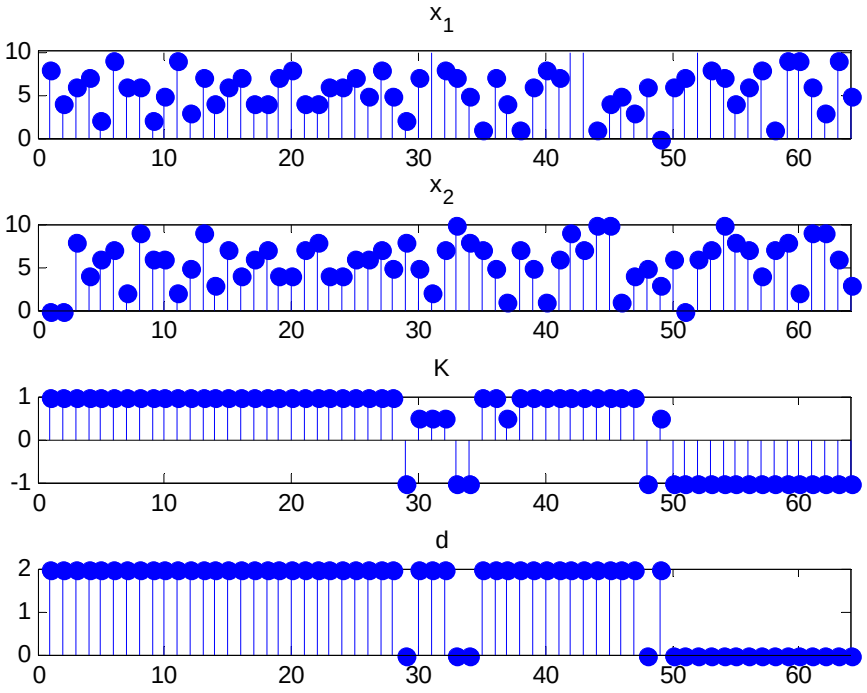


Figure 30. Illustration of the Back Matching Strategy on Random and Noisy Vectors.

3. Computational Complexity and Optimization

The computational complexity of window based matching methods can be very large. If a reference window has dimension w in rows and columns, then from a reference image of M_1 and N_1 rows and columns, respectively, there are $(M_1 - w)\cdot(N_1 - w)$ such reference windows in the reference image. Also, using the same sized target image and target windows, there are $(M_2 - w)\cdot(N_2 - w)$ target windows for comparison. If each reference window is tested against all possible target windows, the complexity is $\Theta\big((M_1 - w)(N_1 - w)(M_2 - w)(N_2 - w)\big)$ where M_1 and N_1 are the rows and columns of the reference image, respectively, and M_2 and N_2 are the rows and columns of the target image, respectively. This complexity is often excessive for a normal set of images, and a reference window need not be tested against every possible window in the target image.

Proper optimization of the algorithm significantly reduces the computational complexity into a manageable subset of necessary comparisons. First, down-sampling the reference and target images reduces the number of rows and columns in each image and reduces the number of reference and target windows that need to be compared. Thus, the complexity at any level is $\Theta\left(\left(\frac{M_1}{r^{l-1}} - w\right)\left(\frac{N_1}{r^{l-1}} - w\right)\left(\frac{M_2}{r^{l-1}} - w\right)\left(\frac{N_2}{r^{l-1}} - w\right)\right)$ where r is the down sampling rate, w is the dimension of the window and l is the level being evaluated. This is due to the fact that an M by N image down-sampled l times by r has dimensions $\frac{M}{r^l}$ and $\frac{N}{r^l}$. Second, applying a search range such that a candidate match can only exist within a search range D around the reference window reduces the complexity to only the target windows within the search range D that need to be checked for each reference window. The resulting optimized complexity for down-sampling and applying a search range is simplified to $\Theta\left(\left(\frac{M_1}{r^{l-1}} - w\right)\left(\frac{N_1}{r^{l-1}} - w\right)D\right) = \Theta\left(M_1' N_1' D\right)$. Ideally, the search range D is chosen to account for the maximum expected disparity

(translation of any point in space between the reference and target images). The search range is reduced from the finest resolution at the same rate as the down-sampling of the images to form the Gaussian pyramid. For example, if a coarse level is half the resolution in rows and columns of its next finest level, the search range within the coarse level should be only half the search range of that at the next finest level. A search in the target image for any disparity greater than the maximum disparity offers no additional information useful for constructing the solution for any level.

Back matching increases the complexity of solving the correspondence problem by multiplying the complexity by at least two. This complexity is increased even more if a top *set* U of matches (the matches depth) are tested in the reverse direction with each candidate match independently tested in the reverse direction for an overall algorithm complexity of $\Theta(MNDU)$ where M is the rows of the image, N is the columns of the image, D is the search range for each reference window and U is the set of candidate matches that need to be back tested for each reference window. In practice, the number of top candidate matches that need to be tested in the reverse direction is usually no more than three since the majority of successful back matches are resolved in the first or second attempt. Failure to find a reverse match by the third attempt usually indicates that no match for the reference window exists in the target region. If a candidate match is found to be within tolerance in the reverse direction, the remaining candidate matches are not tested to reduce computational complexity.

Another computational complexity of note is the memory requirements of the traditional method. Because of the high matching noise as a result of the back matching strategy and the need for error correction, the entire solution for a level in the Gaussian pyramid must be computed before spatial error correction can begin. Error correction is discussed later in the chapter. Thus, in addition to the memory storage of every image within both Gaussian pyramids, the solution at every level must also be stored.

The memory requirement of the Gaussian pyramid of whole images is $4/3$ times the memory storage of the highest resolution. This comes directly from the geometric series of the Gaussian pyramid decomposition where $r = 2$ for each dimension. Since the

solution matrix is of the same dimension of each corresponding level in the pyramid, the solution occupies a proportional amount of memory to one of the stereo Gaussian pyramids.

B. ORDINAL LIMITATIONS WITH TRADITIONAL MATCHING

It has been shown that ordinal measures are very robust for pattern matching. A good pattern matching algorithm is the heart of a stereo matching algorithm that seeks to solve the correspondence problem for computer vision. Yet, the ordinal only describes how matched two patterns are. Thus, untextured or low texture regions of images with low frequency content can appear to have no discernible true match to a window based pattern matching scheme. This fact suggests that although the ordinal, a window based metric, is highly non-linear, it does have a minimum reliable frequency response. For example, given two blank images with no frequency content and no features at all, the depth map between the two images is undefined. Furthermore, pattern matching can also be unreliable in highly uncorrelated patterns. Given a reference and target window with no consistent information between the two, the measure of the patterns should ideally be perfectly uncorrelated. In practice, uncorrelated inputs can appear to have some matching behavior, which can appear to the ordinal as a potential match.

1. Untextured and Low Texture Image Regions

Untextured and low texture comparisons can produce unexpected results for window based matching schemes. A pattern matching algorithm with a set of windows that have no significant information for matching yields inconclusive results. In the case of the ordinal, comparing untextured data appears to be a perfect match, which can mislead the decision process. An example of untextured inputs x_1 and x_2 and the point by point comparison $\kappa\left(x_1\left(n:n+w-1\right), x_2\left(n:n+w-1\right)\right)$ is shown in Figure 31.

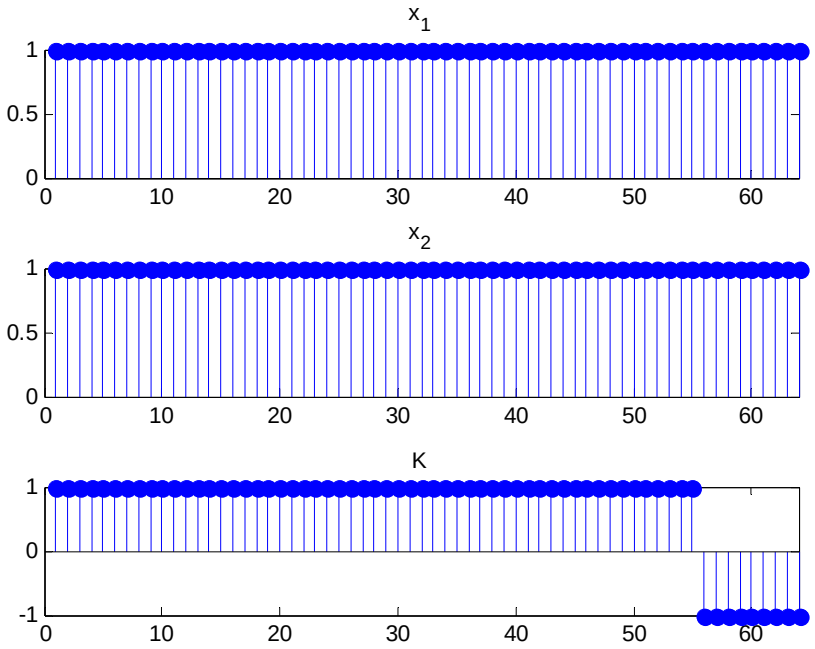


Figure 31. Example of Untextured Inputs and Resulting Point by Point κ.

Applying the back matching strategy does not help the matching of untextured patterns. This is illustrated in Figure 32 of totally untextured signals and Figure 33 with a shifted step where only the step itself has valid data.

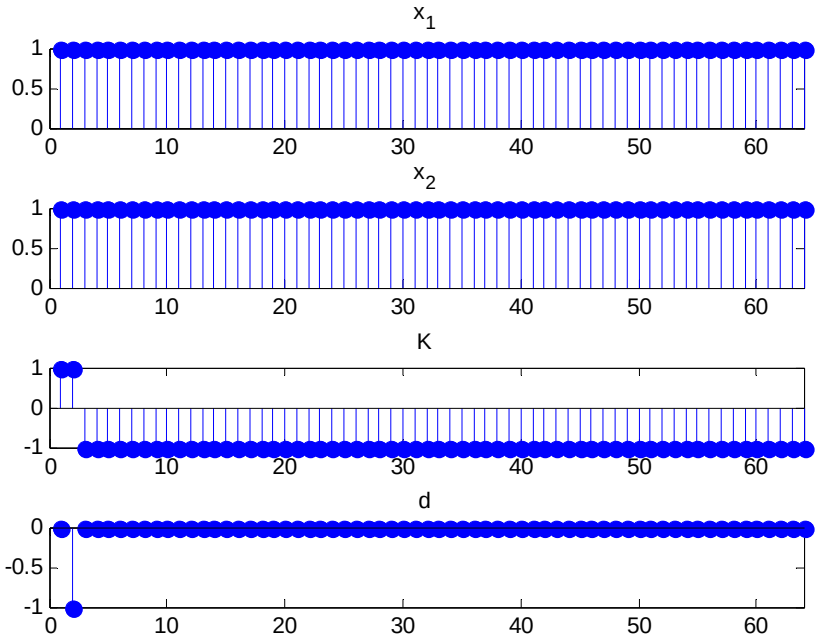


Figure 32. Back-Matching Strategy on Totally Untextured Signals.

The first two values of the κ and disparity vectors result from the forward matching pass choosing the first three matches in x_2 since all possible matches from x_1 to x_2 are equal. In the backward matching pass, each back match chooses the first index in vector x_1 as the top match. Thus, the first disparity is the first sample in x_1 matched to the first sample in x_2, and the second disparity is the second sample in x_1 matched to the first sample in x_2 again. All subsequent disparities fail to find a match because they all continue to choose the first index in x_1 as the top back match but exceed the constraint of being within one sample of the original sample in x_1.

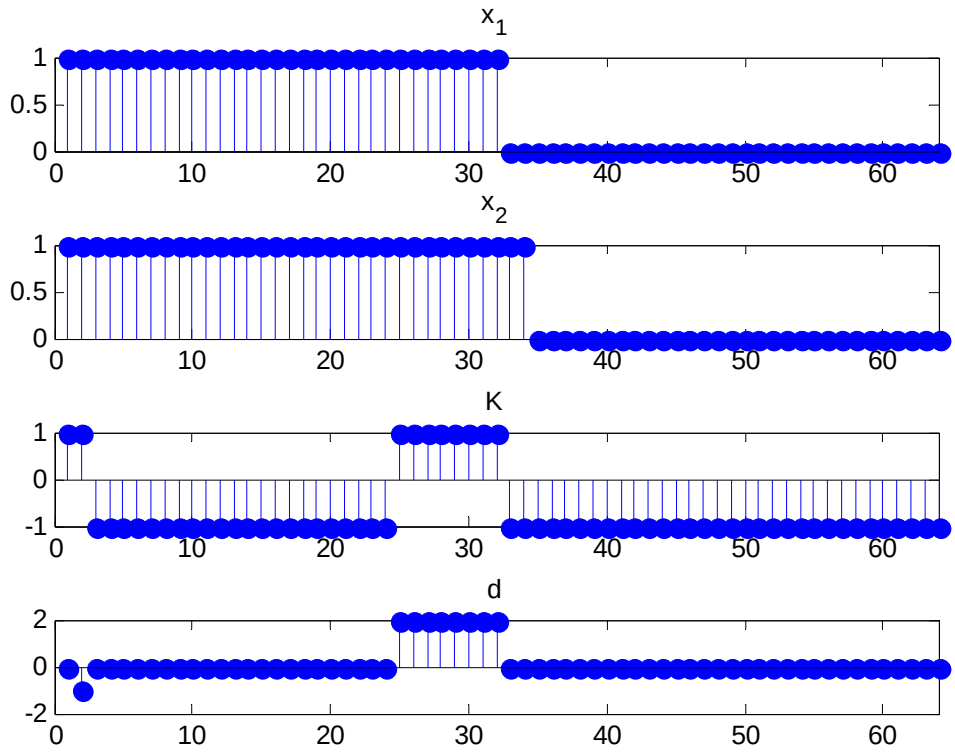


Figure 33. Back-Matching Strategy on Shifted Step in Untextured Data.

The shifted step provides some valid detail for the matching algorithm to match. The step is detected so long as the step is within the window. The samples before and after the step are unmatched due to lack of significant matchable detail. The algorithm correctly determines the step to have translated two samples to the right for a disparity of plus two.

The basic approach to mitigating the effect of untextured matching is to avoid processing reference windows that do not have significant enough details necessary for matching. In this research, a basic measure of window information R, is given by

$$R = \frac{1}{n-2}\sum_{i=0}^{n-2}\left(u(i)-\bar{u}\right)^2 \quad \text{where} \quad u(n) = x(n) - x(n+1). \tag{15}$$

This measure, resembling the variance of the derivative, is applied and thresholded (i.e., compared to an arbitrarily determined minimum measure of information). If the information in the window is above a minimum measure of information, the matching and back-matching algorithm is allowed to proceed. Otherwise, the algorithm skips the reference window and records a disparity of zero and correlation coefficient of -1. The missing solution is accounted for later by error correction methods. A minimum amount of information of 0.05 is chosen for window sizes of 81 samples.

2. Uncorrelated Regions

Uncorrelated inputs can be another problem for window based matching methods and is directly related to the chosen metric's discrimination of dissimilar patterns. The ordinal performs well in discriminating uncorrelated patterns but does still report correlation even when the signals are truly independent. Two possible causes for uncorrelated inputs are occlusions and image edge effects.

Uncorrelated inputs commonly produce a moderate level of correlation but rarely are perfectly matched. An example pair of input vectors independently and randomly generated is input into the algorithm, and the resulting outputs are shown in Figure 34.

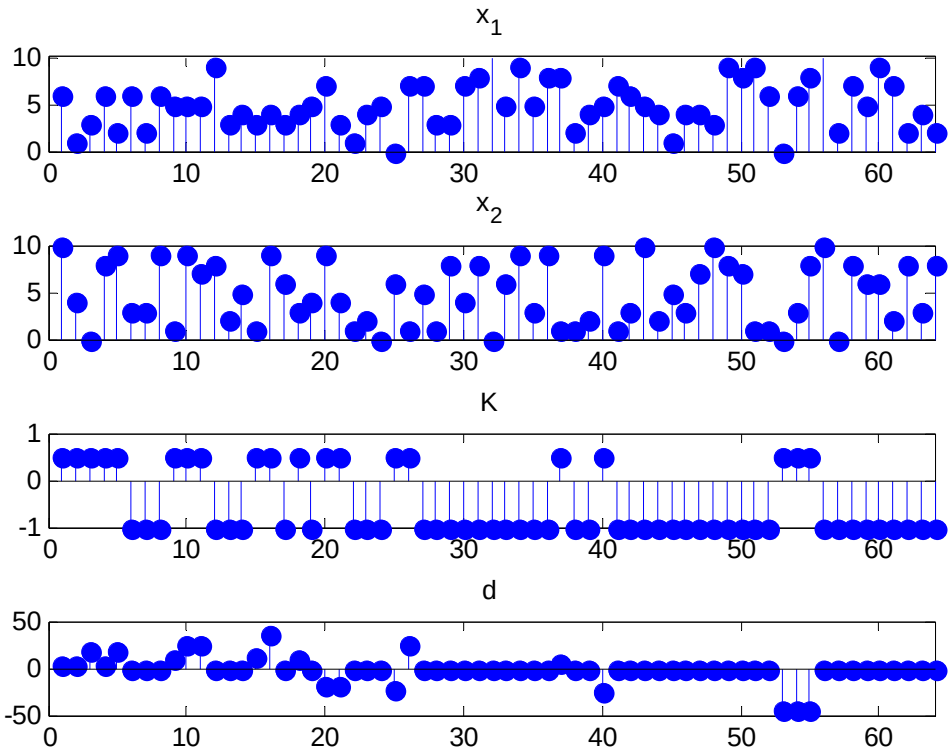


Figure 34. Back-Matching on Uncorrelated Inputs.

Generally, uncorrelated inputs produce a maximum κ correlation of 0.5. Thus, a threshold of the κ values themselves to be at least 0.5 in order to be considered a valid match candidate helps reduce the chance that uncorrelated and unrelated patterns do not factor into the final solution. This number concurs with tests performed by Bhat and Nayar [11] on random dot stereograms with no correlated regions.

C. TRADITIONAL AD-HOC ERROR CORRECTION TECHNIQUES

Since the greedy method introduces significant opportunity for error and that error can propagate and multiply from one level in the Gaussian pyramid to the next, error correction at the conclusion of each level is necessary to improve the quality of the final disparity maps [21]. Several techniques for error correction are considered including preventive techniques such as avoid matching windows with minimal information and recovery techniques such as masked average and linear hole filling.

1. Window Information, Search Ranges and Median Filtering

As described before, window information is measured with Equation (13). When the algorithm computes the minimum information and determines not enough information for matching exists, it stores a correlation value of -1 indicating the need for correction.

Search ranges also help to prevent error by avoiding searches for matches outside the maximum possible disparity at any given level. Since patterns can repeat in images, limiting the search range reduces the chance of choosing a false match within a repeating pattern.

Another effective technique for mitigating error is to median filter the solution using windows of seven by seven or nine by nine. A median filter sorts the data (disparity in this case) from least to greatest and picks the median value. Median filters are non-linear filters that reduce outlier noise. The median value is stored at the centroid of the median filter. Median filter is done after the solution has been computed but before any further ad-hoc error correction.

2. Masked Averaged Hole Fill

Masked averaged hole filling corrects the disparity map by determining invalid regions in the disparity map and filling the "holes" formed by the invalid data with immediately adjacent valid data. The segmentation generates a mask of values that need to be correct through spatial correlation. The holes are filled iteratively using a block structuring element to propagate valid data into the invalid region and a single erosion of the mask using the same structuring element. The process continues until all invalid data indicated by the mask is eroded away. Thus, the masked averaged hole fill algorithm has a theoretical computational complexity of $\Theta\left(\left(MN\right)^2\right)$ if only one valid pixel exists in the entire solution and a lower bound complexity of $\Theta\left(MN\right)$. In practice, the masked averaged hole fill algorithm is close to the lower bound complexity unless the disparity map has substantial error indicated by the mask.

The invalid data is segmented by the threshold of the correlation value of the chosen match for each reference point in the reference image. The threshold is chosen to be 0.5 to avoid accepting matches that could have resulted from noise or uncorrelated inputs. The invalid data also includes points in the solution that are not matched at all because the reference window does not contain enough information. The result is a mask with the same dimensions as the solution, assigning a value of '1' to an invalid solution and a '0' to a valid solution. The mask is passed to the hole filling part of the error correction algorithm, and the output of the algorithm is the corrected solution with only valid data. A block diagram of the hole filling algorithm is shown in Figure 35.

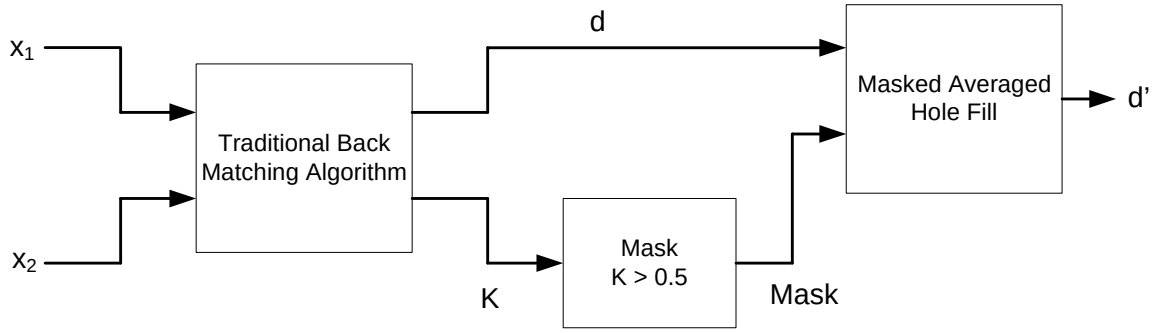


Figure 35. Block Diagram of Ad-Hoc Masked Averaged Hole Fill Algorithm.

Filling the invalid regions with valid data resembles a zero-order hold. Valid solutions are iteratively repeated until the invalid region is filled. If two or more solutions are propagated to occupy the same invalid point, the solutions are averaged. The process is illustrated in Figure 36.

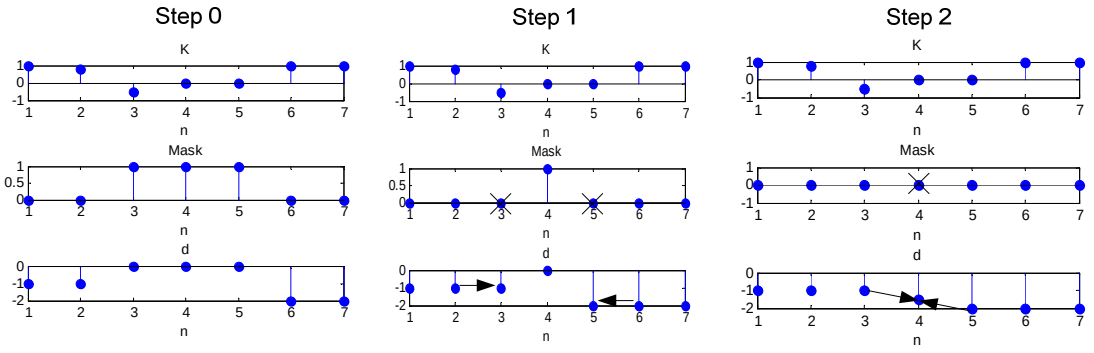


Figure 36. Illustration of Masked Averaged Hole Fill Algorithm.

The masked averaged hole fill spatially corrected solution is passed to the next step in the disparity computation algorithm.

3. Masked Linear Hole Fill

The masked linear hole fill error correction algorithm is similar to the masked average hole fill algorithm, but the valid data propagation is extended to a first order approximation. The invalid data is segmented by applying the 0.5 minimum threshold and assigning a mask bit of '1' to solutions that need to be replaced with the linear approximation. The overall block diagram of the masked linear hole fill algorithm is shown in Figure 37.

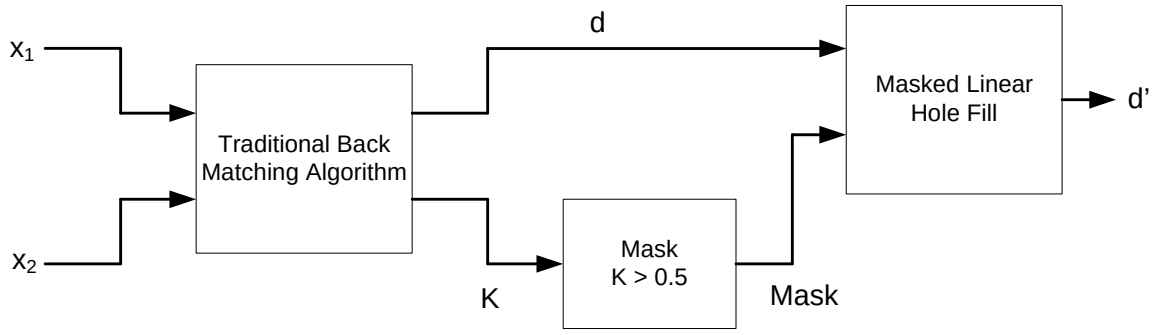


Figure 37. Block Diagram of Ad-Hoc Masked Linear Hole Fill Algorithm.

The linear approximation is computed in the horizontal and vertical directions independently and averaged point by point for each invalid solution indicated by the mask. First, the bounding valid solutions around the invalid space are identified. The algorithm computes the difference between the bounding solutions and divides by the number of invalid solutions plus one representing the linear change between each solution and its immediate neighbor. The invalid solutions are filled by accumulating the change to the previous solution starting from the left valid solution and end at the right valid solution. The same process is repeated for every invalid region in the vertical direction. The process is illustrated in one dimension in Figure 38. The implementation of the masked linear hole fill algorithm has a complexity of $\Theta(MN+k)$ where k is the number of linear corrections needed.

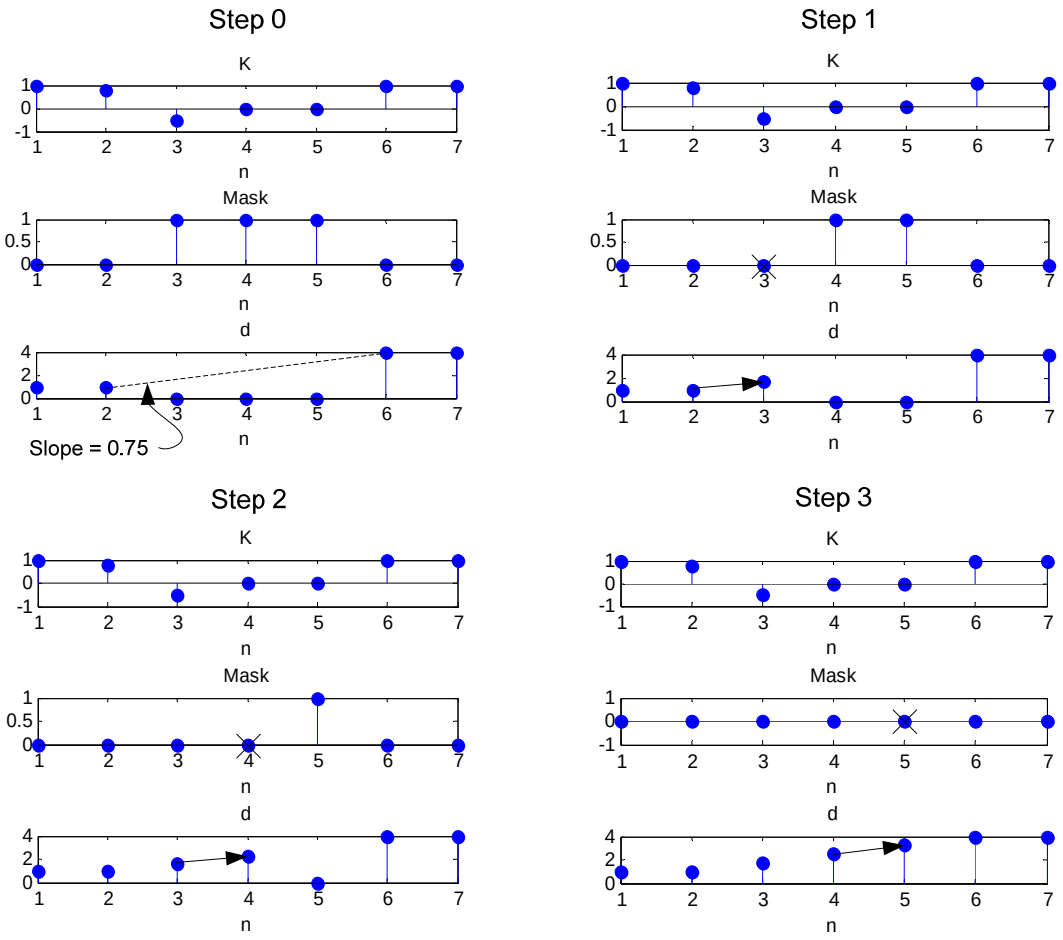


Figure 38. Illustration of Masked Linear Hole Fill Algorithm.

The masked linear hole fill spatially corrected solution is passed to the next step in the disparity computation algorithm.

D. RESULTS

The traditional method solves the correspondence problem with a dense disparity map, providing a solution vector for every point in the reference image. Applying error correction, we discard portions of the dense disparity map and replace the holes with redundant data. Thus, the raw dense disparity map produced by the back matching strategy can be seen as a sparse matrix of valid solutions, and error correction recovers the missing data between valid solutions. The raw solutions alone do not satisfactorily meet the objective of accuracy set out in Chapter I.

1. Raw Back Matching Output

First, the raw output from the back matching strategy is evaluated. For computational reasons, the search range and Gaussian pyramid are used in producing the raw output, and therefore, their impacts are not independently considered. The disparity maps are heavily corrupted with errors, particularly in regions with low information. The raw output using three levels and a search range of $+/-5$ at the coarsest resolution and $+/-1$ at each finer resolution is shown in Figure 39.

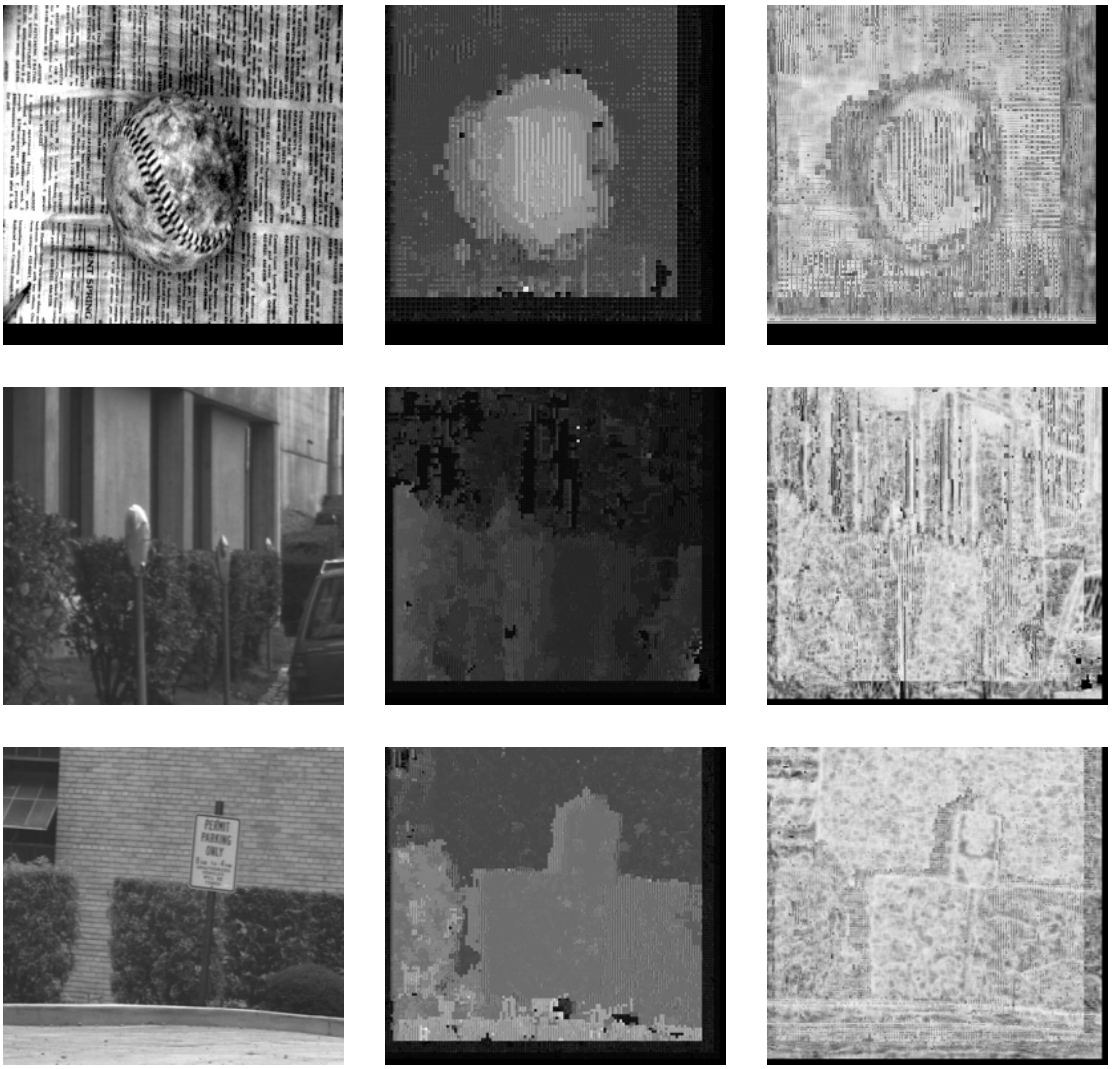


Figure 39. Raw Output of Multiple Images: Ball, Meter and Shrub. After [19], [23], [24]

The intensity at each point is the magnitude of the vector describing the change in x and y for the pattern in the reference image to the matched target pattern in the target image. The associated κ at each point is shown for illustration. Higher intensity κ means that the chosen match at that point was a good one, and low intensity or black mean that the chosen match was weak or the algorithm did not find a match.

The uncorrected error corrupts the disparity maps beyond usefulness. Thus, additional error correction is necessary at each level for the disparity maps to be meaningful for any subsequent image segmentation.

2. Spatially Correlated Correction

a. Masked Averaged Hole Fill With Median Filtering

The masked averaged hole fill algorithm is applied to the resulting raw disparity map solution output from the back matching strategy algorithm at each level. The results are shown in Figure 40 using three levels and a search range of $+/-5$ at the coarsest resolution and $+/-1$ at each finer resolution.

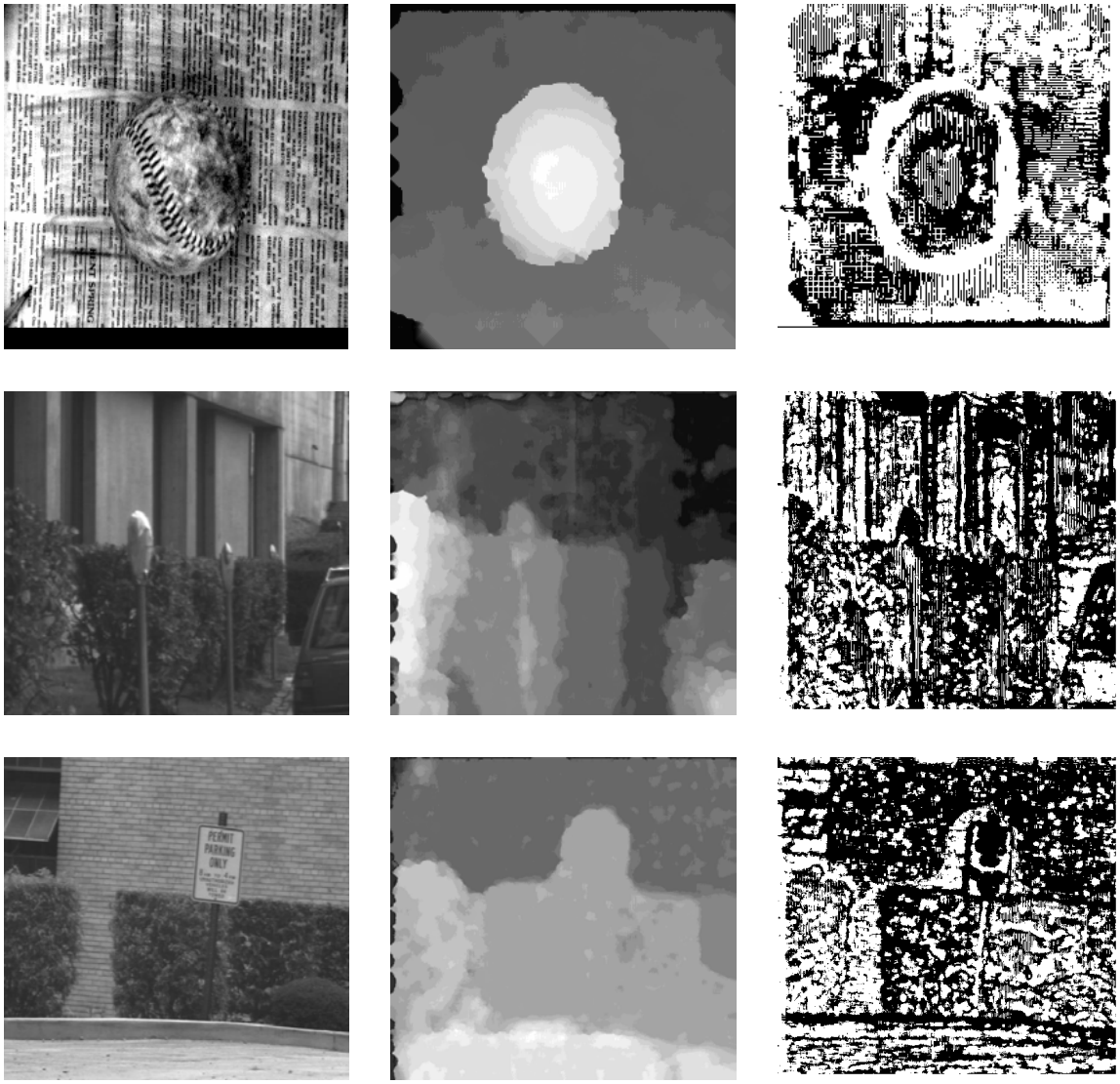


Figure 40. Masked Average Hole Fill Corrected Disparity Maps of Multiple Images: Ball, Meter and Shrub. After [19], [23], [24]

Like before, the intensity at each point is the magnitude of the disparity vector with respect to each point in the reference image pointing to the match in the target image. The associated mask is the result of thresholding the correlation coefficients κ below 0.5, and white pixels indicate where spatial correlation correction was applied. The white pixels in the mask are the discarded disparity solutions that were replaced by the zero-order hold nearest neighbor approach of the masked averaged hole fill algorithm.

One can notice by the mask that significant error exists using the traditional matching techniques, and about half of each disparity map consists of spatially correlated values.

The error corrected disparity maps are cleaner and more consistent. The results illustrate that the traditional method can produce disparity maps with enough quality for object segmentation. However, the *accuracy* of the disparity maps is quite low. For example, the sign is certainly distinct from the background, but the edges determined by the masked averaged hole fill algorithm are quite different from the true edges of the sign in the reference image. This effect can also be seen in the significant uncertainty around the edges of the ball (as depicted in the ball mask) or the near field of the meter mask. Therefore, sharp edges in the disparity map help to segment objects but do not indicate accuracy in the disparity map.

b. Masked Linear Hole Fill With Median Filtering

The masked linear hole fill algorithm is also applied to the resulting raw disparity map solution output from the back matching strategy at each level like the masked average hole fill. The difference is the linear interpolation of the sparse disparity solution matrix after masking to a dense disparity solution matrix. The results are shown in Figure 41 using three levels and a search range of $+/-5$ at the coarsest resolution and $+/-1$ at each finer resolution.

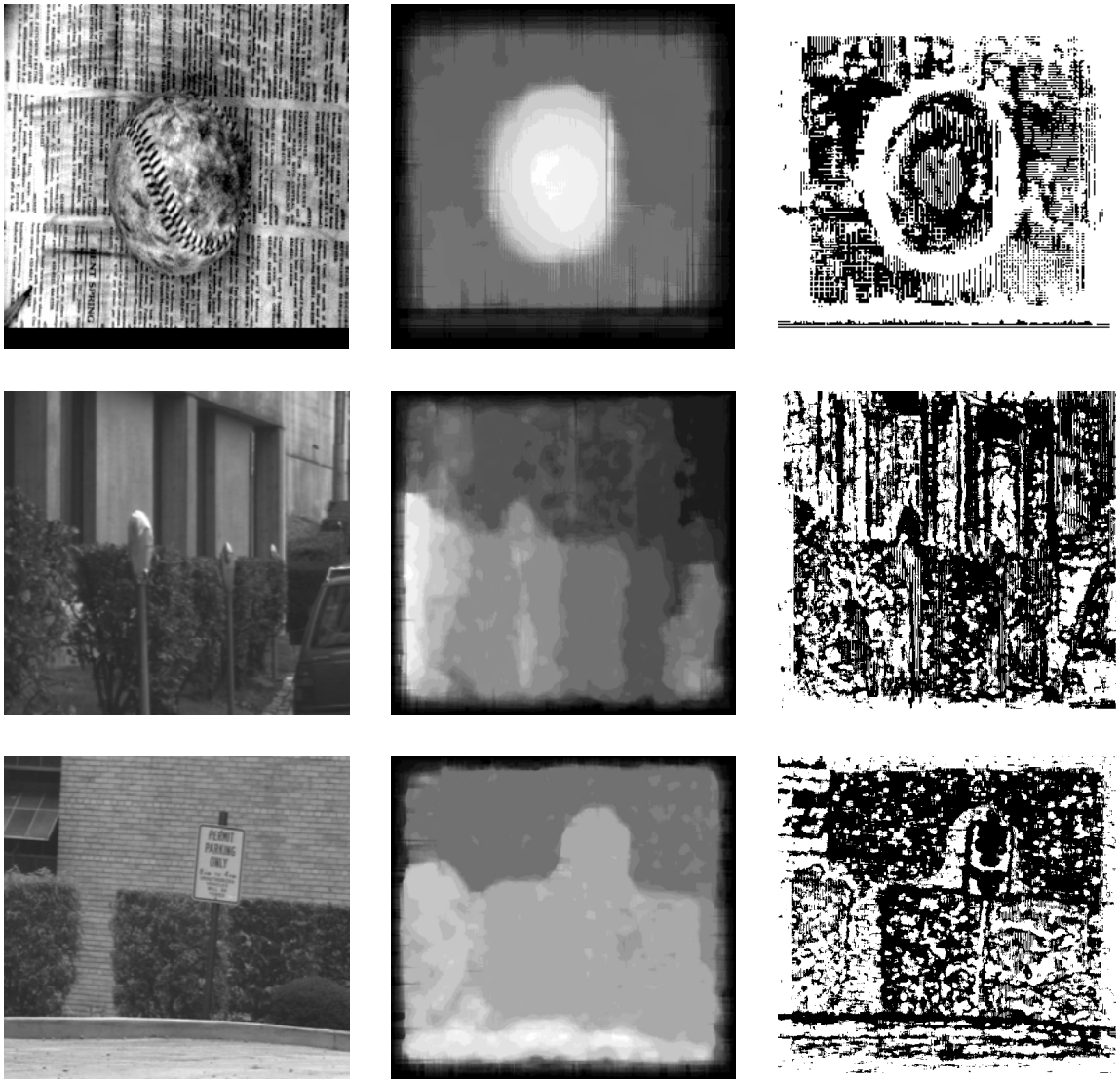


Figure 41. Masked Linear Hole Fill Corrected Disparity Maps of Multiple Images: Ball, Meter and Shrub. After [19], [23], [24]

Like the masked average hole fill results, the intensity of the disparity map indicates the magnitude of the disparity vector with respect to the reference image, and the mask indicates where spatial correction was applied (white pixels). The disparity map solutions are linearly interpolated by the masked linear hole fill algorithm, and the effects can be seen by the slight banding of intensity values around the solutions.

The error corrected disparity maps with the linear interpolation produce cleaner and more consistent disparity maps when the invalid solutions occupy regions of varying depth while having similar impact to the masked averaged hole file in regions of constant disparity. Thus, the masked linear hole fill algorithm generally does a better job of error correction than the masked averaged hole fill. However, the *accuracy* is still quite low. The edges in the disparity map are blurred and seem to more accurately depict the depth process in the ball image, but the masked linear hole fill algorithm fails to accurately determine the depth process in both the shrub and meter disparity maps much like the masked averaged hole fill algorithm.

3. Computational Complexity and Memory Requirements

The traditional back-matching strategy is not particularly computationally efficient. For the raw results alone, the computational complexity is high even with the Gaussian pyramid sub-sampling with a growth function $\Theta(MNDU)$ where M and N are the rows and columns of the reference image, D is the search range and U is the set of matches to be back-matched. Employing the masked averaged hole fill algorithm could add $\Theta\left((MN)^2\right)$ between each level in the pyramid. The masked linear hole fill algorithm adds $\Theta(MN+k)$ complexity where k is the number of corrections to be made between each level in the pyramid. Thus, employing the masked linear hole fill algorithm for error correction would have a total computational complexity of $\Theta\left(MNDU+(MN+k)\cdot L\right)$ where L is the number of levels. Also, the nature of the traditional back-matching strategy makes the true computational complexity difficult to predict.

The memory requirement for the traditional algorithm is approximately three times the memory occupied by the original stereo pair. This accounts for both full-sized Gaussian pyramids along with full-sized intermediate solutions. Full-sized intermediate solutions are needed for the ad-hoc error correction to function properly. Also, the precise memory requirements during run-time are unpredictable because the ad-hoc error correction depends on the number of invalid solutions indicated by the error correction mask.

The computational cost and memory requirements running the traditional matching technique with a nine by nine window, three levels and $+/-5$ disparity at the coarsest levels and $+/-1$ disparity at each finer resolution are shown in Table 2. All experiments were performed on a computer equipped with an Intel Core 2 Extreme X9000 processor and 4 GB of available RAM. The peak memory requirements are also shown (including the buffering of the stereo pair of images). These measurements were taken using Microsoft Windows Task Manager and are shown only for illustrative purposes.

Table 2. Traditional Matching Timing and Memory Requirements.

Stereo Pair	Processing Time	Peak Memory Usage
Ball (256 x 256)	30 s	2976 kBytes
Meter (512 x 480)	111 s	9684 kBytes
Shrub (512 x 480)	127 s	10584 kBytes

The processing time is approximately linear with respect to the size of the input images, but the memory requirements can vary significantly depending on the amount of error correction needed. Also, a significant portion of the processing time (about 95%) was dedicated to calculating the correlation coefficient κ even with a fully optimized C implementation.

The traditional method was investigated and demonstrates the ability of the ordinal to solve the correspondence problem by the greedy method. In the next chapter, a more efficient dynamic programming algorithm with better memory requirements is investigated. Dynamic programming offers improved matching, complexity and memory requirements.

V. ORDINAL MATCHING WITH DYNAMIC PROGRAMMING

In recent years, dynamic programming techniques have been applied to the correspondence problem, often producing better results than the traditional greedy method of matching [12], [13], [25]. Dynamic programming computes solutions to the correspondence problem to optimality given constraints on smoothing and spatial features of the depth process. Thus, unlike the traditional method, dynamic programming can account for the disparity of a set of points simultaneously before committing to a solution. This advantage of dynamic programming reduces the need for error correction to produce good quality disparity maps.

Dynamic programming is often applied to solve stereo correspondence problems reduced to one dimension. Those algorithms begin by rectifying the scan lines of each stereo image so that the scan lines coincide with the epipolar plane formed by the stereo pair of cameras. Rectification requires information about the stereo pair of cameras, which means that such approaches are not a general solution to the correspondence problem but a specific one. Therefore, a unique method of two-dimensional dynamic programming particular to window based methods is investigated that retains general solution qualities.

A. DYNAMIC PROGRAMMING MATCHING STRATEGY WITH THE ORDINAL

Changming Sun [12] illustrated a dynamic programming approach to window based correlation methods similar to the ordinal. His algorithm was shown to be very efficient, solving the correspondence problem in milliseconds on a typical personal computer and producing results comparable to the graph cuts method (very accurate but one of the most computationally expensive and dynamic approaches). Sun's dynamic programming method can be adapted to use the ordinal instead of the zero-mean normalized cross co-variance chosen for his research.

The dynamic programming method centers on the construction of a solution volume where the disparity map is the surface with the maximum correlation within the

volume. While the solution volume can be extended to hyper dimensionality to account for two dimensional disparity vectors, the solution volume is instead constructed for only disparity along the scan lines (horizontal translation). The consequence of this simplification is that as the epipolar plane deviates from the scan lines, the accuracy of the matching algorithm falls off dramatically. This can occur if one camera in the stereo pair is out of alignment with respect to the other camera rotationally, vertically translated or both. For this research, the stereo pair is assumed to be in sufficient alignment that the epipolar plane is consistent enough for successful window based matching along scan lines.

1. Multirate and Sub-Regioning With a Pyramid

Like the traditional back-matching method, Sun's algorithm uses multirate to reduce computational complexity and improve matching accuracy. The down-sampling procedure is simplified from the Gaussian pyramid down-sampling using the follow steps:

1. Given a sub-region of M_i x N_i pixels and r down-sampling rate, the sub-region is divided into non-overlapping r x r blocks.
2. The down-sampled sub-region is the simple average of each r x r block.

The dynamic programming method proposed by Sun also introduces sub-region partitioning to improve computational complexity. Each sub-region is chosen to minimize the complexity $\Theta(M_i N_i D_i)$ where M_i is the rows of sub-region i, N_i is the columns of sub-region i and D_i is the expected maximum range of disparities within the sub- region i. The partitioning results from a divide-and-conquer scheme where an initial sub-region partitioning is refined by merging regions with similar complexity and dividing larger regions into smaller regions with less cumulative complexity. Therefore, the goal is to divide the level in the pyramid into small regions of high deviation disparities and larger regions with low deviation of disparities. The sub-region partitioning is known to be less complex than $\Theta(MND)$ where M and N

are the rows and columns respectively of the un-partitioned level in the pyramid and D is the maximum disparity range of the level.

Partitioning the working pair at a level in the pyramid into sub-regions constitutes the first step to constructing a volume of solutions. For a particular sub-region i taken from the reference and target images, the solution volume is calculated by shifting the target sub-region through the range of disparities $-d_{\max}$ to $+d_{\max}$, where $[-d_{\max},\ldots,d_{\max}]=D,$ with respect to the reference sub-region. For each point in the reference sub-region, the associated correlation values at each disparity d are recorded in the depth of the volume. The process is illustrated in Figure 42.

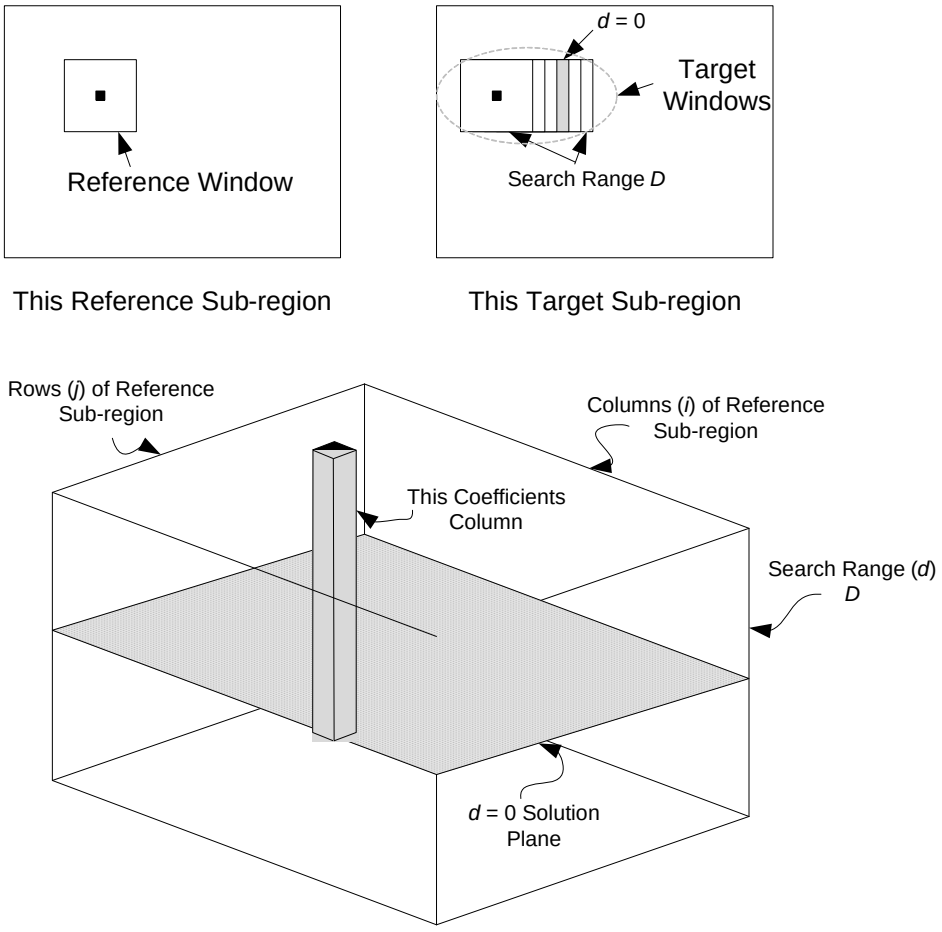


Figure 42. Illustration of Constructing the Solution Volume $C(i,j,d)$.

At each level l where the base of the pyramid is level zero, the optimal surface of the disparity is calculated and propagated to the next level $l-1$ in the pyramid to constrain the following search to within the neighborhood of the previous target match. The disparity map of the previous level $l+1$ is up-sampled and multiplied by the rate used to construct the pyramid. After up-sampling, the disparity map is linearly interpolated to reduce discontinuities. The coarse to fine algorithm is illustrated in the following steps:

1. Beginning with the top level, partition the current level l into sub-regions of minimum complexity.
2. For each sub-region, compute and solve the solution volume with dynamic programming using the up-sampled disparity map of level $l+1$ as the initial search. If the current level is the top, the initial disparity map is assumed to be all zero.
3. Up-sample the level l solution by linear interpolation and multiply by the sampling rate.
4. Return to step 1 if level $l \neq 0$.

2. Solving the Solution Volume With Two-Stage Dynamic Programming

After the solution volume is constructed, the maximum surface through the volume is computed using a two-stage dynamic program (TSDP). The maximum surface is the optimal path based on the maximum summation of correlation coefficients and the constraint p on the maximum slope of the surface in either the vertical or horizontal direction. If the dynamic program is not given a constraint, it selects the maximum correlation values at each point in much the same way matches are chosen in the greedy method explained in the previous chapter. The maximum surface is illustrated in Figure 43 and represents the disparity of the sub-region.

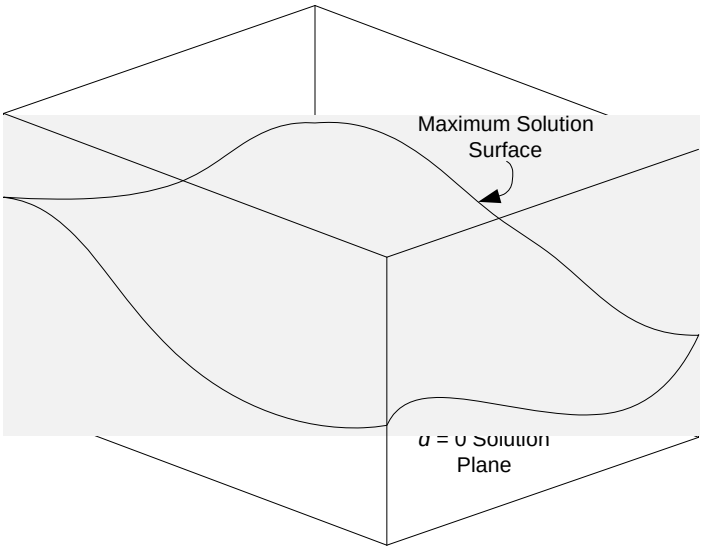


Figure 43. Maximum Surface through the Solution Volume. After [12]

The first stage in solving for the maximum surface is to compute the summation of correlation values as a cost function from the solution volume. The summation volume is $Y(i, j, d)$ where i and j are the rows and columns of the volume and d is the index through the range of disparities for each row and column. The summation volume is calculated using the recursive formula

$$Y(i, j, d) = C(i, j, d) + \max_{t:|t| \le p} Y(i, j-1, d+t). \tag{16}$$

The first stage of the dynamic program proceeds by selecting a column by disparity slice of the solution volume. Then, the dynamic program computes the optimal summation of cost functions from the top row to the bottom row in the slice for each disparity. This process is illustrated in Figure 44.

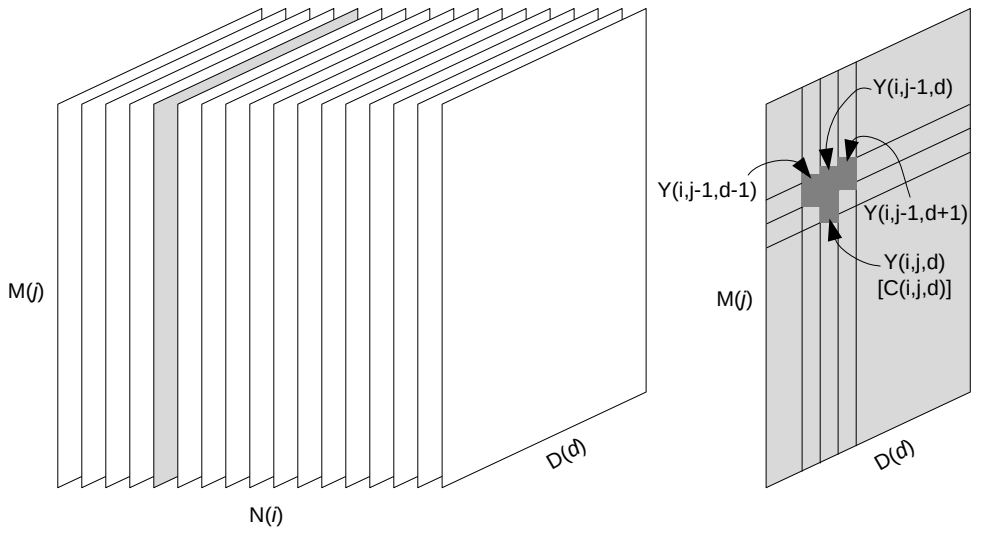


Figure 44. Computing the Solution Volume by Slice. After [12]

The second stage of the dynamic program begins by finding the optimal path through the row by disparity slice. The optimal path is the minimum distance recursion through the bottom slice of the summation volume $Y(i, j, d)$. The minimum distance $G(i, j, d)$ recursion is defined by the recursive equation

$$G(i, j, d) = Y(i, j, d) + \max_{t:|t|\le p} \{G(i-1, j, d+t)\}. \quad (17)$$

Once the optimal path through the bottom slice is determined, the second stage proceeds to the next slice (row) up of the summation volume. Subsequent slices' optimal paths above the bottom slice are constrained to be within p disparity of the previous optimal path. The second stage dynamic program is illustrated in Figure 45. Thus, the minimum distance recursive algorithm for calculating the optimal path through the i^{th} slice of the summation volume is:

1. Compute the distances using Equation 17 starting from $i = 1$ and computing the maximum summation given the slope constraint p.
2. Record the back-pointer to each maximum t^{th} (as in Equation 17) cost summation.
3. After the minimum distance recursion, choose the final maximum cost.
4. Recurs backward using the back-pointers to form the optimal path.

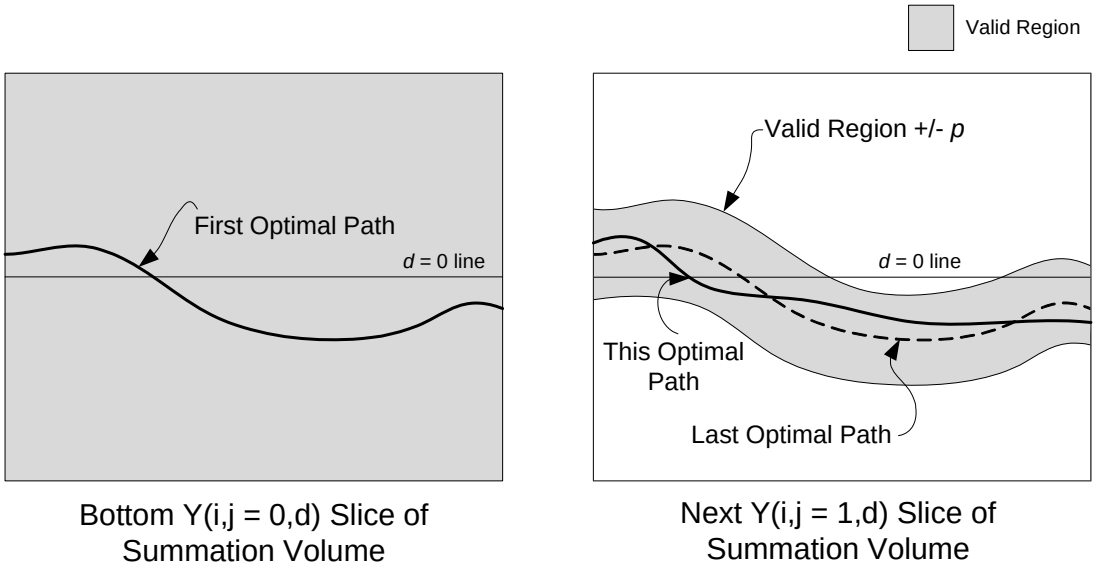


Figure 45. Computing the Optimal Path through the Summation Volume. After [12]

Finally, the optimal paths are concatenated into a surface of disparities. The resulting surface is the optimal solution of the solution volume. This solution is up-sampled, interpolated and multiplied by two if the solved level is not the base of the pyramid to constrain the search range of the next finest resolution.

B. OPTIMIZATIONS

The dynamic programming approach to solving the correspondence problem allows for more freedom in orienting the problem to be spatially (memory) and computationally efficient. While Sun's implementation of the TSDP is efficient for software, further optimizations are necessary to make the algorithm more feasible in hardware. Hardware efficiency is maximized when the algorithm uses a constant space and computational complexity. This is because hardware must always account for the algorithm's worst-case scenario in computational complexity. Anything less than worst-case complexity leaves some hardware logic idle, resulting in an inefficient implementation. Thus, the sub-region partitioning and construction of the solution volume are modified to optimize hardware computational complexity.

1. Constant Sub-Region Partitioning

The sub-region partitioning is essential for breaking down the stereo pair into manageable bits and preventing the solution volume from becoming overly ponderous. However, variable sub-region partitioning requires varying memory requirements, which can cause problems for an efficient hardware implementation. Thus, sub-region partitioning is refined such that sub-regions have constant dimensions.

Optimization of sub-region partitioning for hardware has a couple key differences from Sun's method. First, sub-region partitioning can be adapted to operate on different parts of the stereo pair independently. This can dramatically reduce the working memory requirements of a hardware implementation since the algorithm must only buffer the regions within the reference and target image of interest. Therefore, the original stereo pair is partitioned into constant dimension sub-regions, and the pyramid decomposition is performed on the partitions and not the original stereo pair. Second, the sub-regions are kept to a constant square dimension a (with overlapping edges in the target sub-region to account for search range edge effects) so that computational complexity is kept constant. The algorithm for constant sub-region partitioning is defined using the following steps:

1. For the reference image, divide the image into $a \times a$ non-overlapping sub-regions. For the target image, divide the image into the same $a \times a$ sub-regions but with additional padding D to account for search range.
2. For each sub-region, decompose the sub-region into a pyramid.
3. Solve the solution volume proceeding from coarse to fine.
4. Insert the disparity map of the finest resolution into the final solution.
5. Repeat step two until all sub-regions are solved.

The sub-region partitioning is illustrated in Figure 46. In this example, the target sub-region includes padding to the left and right. In practice, it is only necessary for the overlapping padding in the target to be on the right side if the reference image is the right stereo image and on the left side if the reference image is the left stereo image for simplicity.

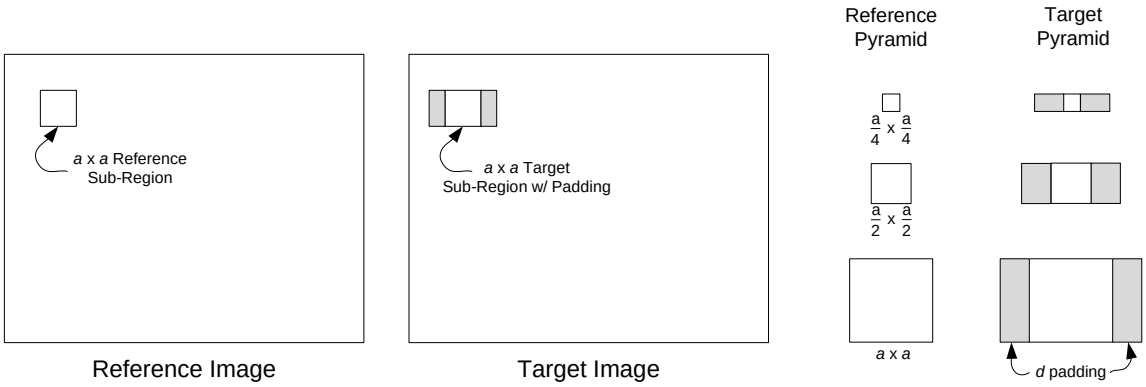


Figure 46. Example of Constant Sub-Region Partitioning and Pyramid.

2. Abstract Solution Volume

Since the sub-regions are now partitioned constantly, generating a solution volume for all possible disparities at each level in the pyramid becomes impractical. The cubic shape of the solution volume grows to maximum complexity at the base level for nearly every sub-region. Therefore, the solution volume is redefined such that the volume complexity is always the minimum possible $\Theta(M_l N_l D)$ where M_l is the rows of the constant sub-region at level l and N_l is the columns of the constant sub-region at level l and D is the uniform search range at each level.

The minimum solution volume is achieved by forming an abstract volume of solutions around the solution surface of the previous level. After the previous solution at the $l+1$ level is up-sampled and interpolated, a search range is constructed around each point in the up-sampled solution surface to form a new solution volume around the $l+1$ surface. This simplification in complexity is intuitive since the solution at level l must always be within the neighborhood of the $l+1$ level solution. The new solution volume construction algorithm is defined in the following steps:

1. For each pixel in the reference sub-region, construct a reference window.
2. Translate to the within the target neighborhood of the previous solution and compute the correlation values for D disparities.
3. Record the correlation values and associated cumulative disparity d in the abstract solution volume.

The abstract volume can be efficiently implemented using C++ techniques. The important innovation that makes the abstract volume possible is the recording of the disparity along with the correlation value and always D disparities for each reference pixel. This results in a constant $\Theta(M_I N_I D)$ complexity with proportional memory requirement. To illustrate this, the abstract volume is compared to the cubic volume in Figure 47.

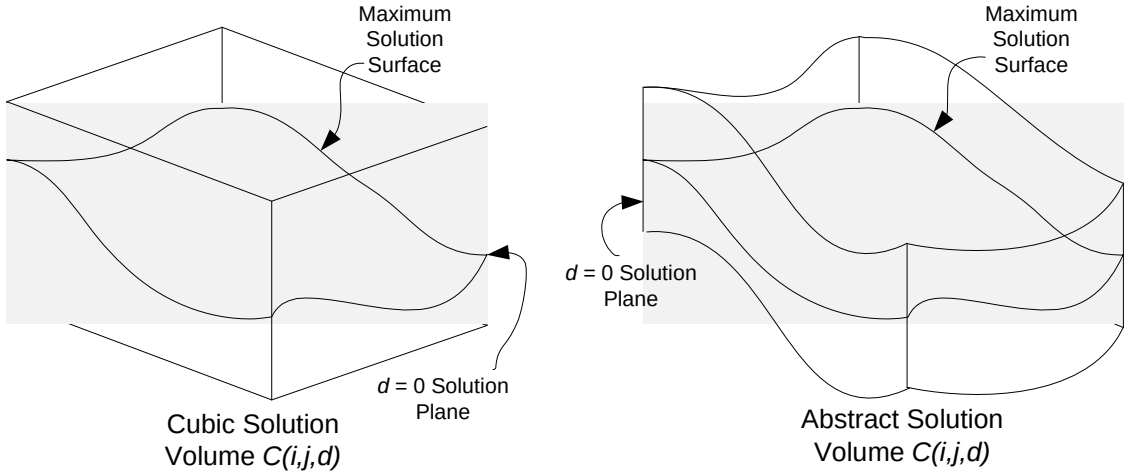


Figure 47. Visualization of the Abstract Volume Compared to the Cubic Volume. After [12]

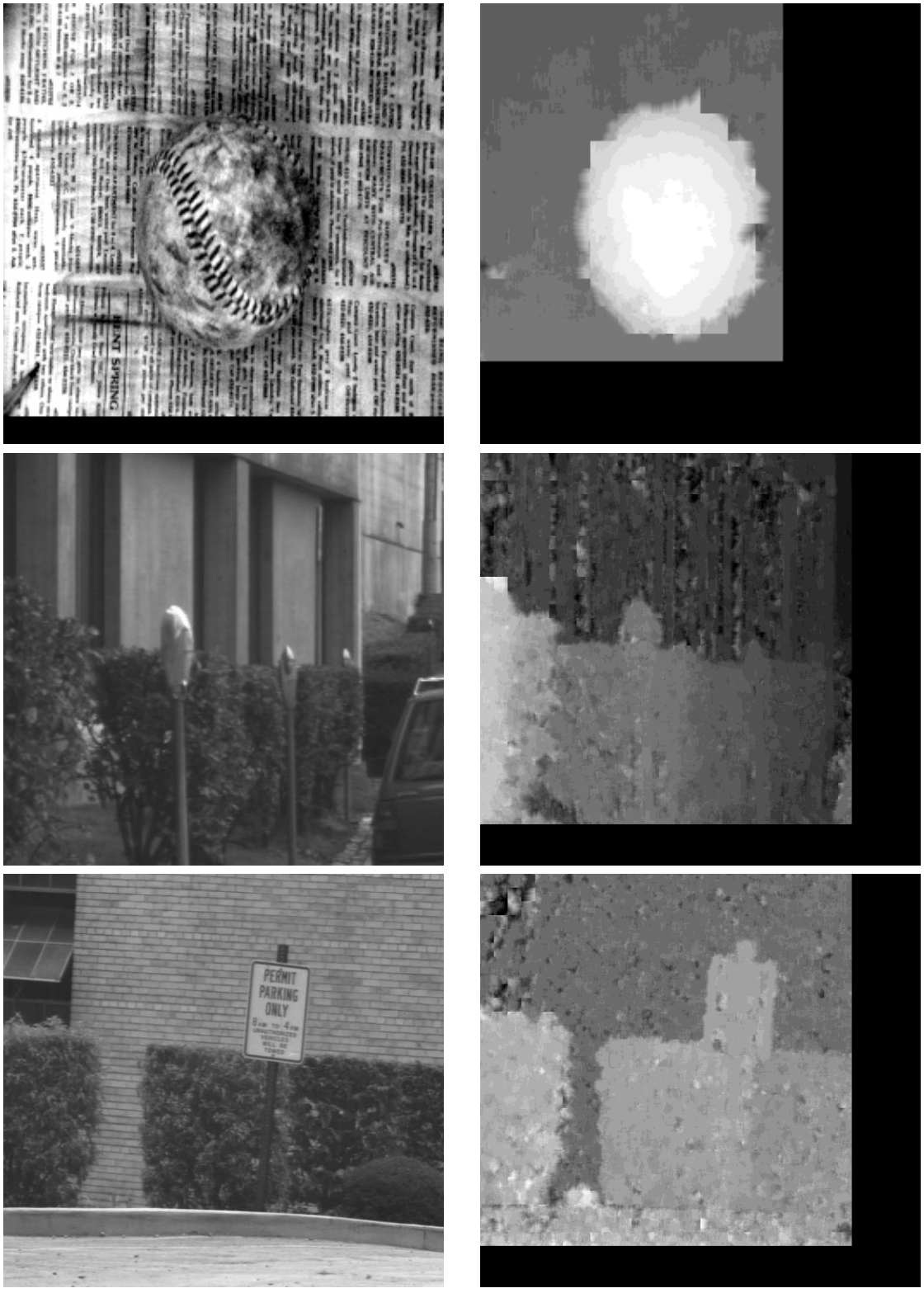
The abstract solution volume is stable so long as the maximum slope p of the solution surface does not exceed the search range D. If the slope is greater than the search range, the abstract solution has discontinuous regions that cannot be solved using the second stage minimum distance of the dynamic program. The discontinuity appears to the TSDP algorithm as all paths through the discontinuity having infinity cost.

C. RESULTS

1. Raw Dynamic Programming Matching Output

The dynamic programming algorithm yields disparity maps of good quality without any error correction. For these experiments, the number of levels is set to three, window dimension is 11 by 11, sub-region dimension is four by four and the search range

is four. Due to an implementation issue with the dynamic program search function, the sub-region dimension cannot get too large or very little of the image will be available for processing due to edge effects. Thus, the sub-region dimension is kept small, leading to potential for moderate inter-sub-region error. Also, the large black regions on the right and bottom of the images are due to the edge of effects of a weak sub-region implementation. The disparity maps produced by the algorithm are shown in Figure 48. While it is not obvious, the disparity maps produced by the dynamic program are more accurate than any of the disparity maps produced by the traditional back-matching method. This is evident in the clearly defined edges of objects such as the meter and the sign in the meter and shrub stereo pairs where all implementations of the traditional method failed.



Reference Image Disparity

Figure 48. Disparity Maps from Dynamic Programming for Ball, Meter and Shrub. After [19], [23], [24]

Error correction techniques are still possible on the final or intermediate disparity maps. Although not shown, the correlation coefficient of each chosen match is known, and either the masked averaged hole fill or masked linear hole fill algorithms can be applied to spatially correlate disparity values that are below a minimum κ threshold.

2. Computational Complexity and Memory Requirements

The dynamic programming with applied abstract volume and constant sub-region partitioning optimizations produced results that satisfactorily met the objectives of this research. The computational complexity is a constant $\Theta(M_i N_i D)$ where M_i is the rows and N_i is the columns of the chosen sub-region partitioning. It is noteworthy that this computational complexity approaches the theoretical minimum possible for the stereo correspondence problem. The memory requirements are proportional to the computational complexity since only one solution is stored for $\Theta(M_i N_i D)$. Using the ordinal and minimum complexity abstract solution volumes, we get the disparity map computation times for various images and parameters tabulated in Table 3. All experiments were performed on a computer equipped with an Intel Core 2 Extreme X9000 processor and 4GB of available RAM. The peak memory requirements are also shown (including the buffering of the stereo pair of images). These measurements were taken using Microsoft Windows task manager and, thus, are inherently inaccurate and shown only for illustrative purposes.

Table 3. Dynamic Programming Timing and Memory Requirements.

Stereo Pair	Processing Time	Peak Memory Usage
Ball (256 x 256)	26 s	1924 kBytes
Meter (512 x 480)	124 s	4288 kBytes
Shrub (512 x 480)	124 s	4288 kBytes

The processing time and peak memory usage are linear with respect to the size of the input image pair. The peak memory usage was constant since the memory usage is entirely dependent on the size of the sub-region partitioning in this implementation.

Also, a significant portion of the processing time (about 95%) was dedicated to calculating the correlation coefficient κ even with a fully optimized C implementation.

The dynamic programming method was shown to be effective with the ordinal as the correlation metric. In particular, it produced more accurate disparity maps compared to the traditional greedy method without the need for ad-hoc error correction. However, ad-hoc error correction can still be applied to clean up the minor errors in the dynamic programming matching process.

VI. CONCLUSIONS

It was shown that the ordinal is a logical measure of pattern correlation that can be synthesized into hardware. Since with all stereo correspondence algorithms, the pattern matching measure is at the core of all processing, obtaining a fast metric implementation is crucial to achieving real-time stereo processing.

A. SUMMARY OF THE WORK

First, the ordinal was investigated for performance with the traditional back-matching strategy with ad-hoc error correction. The traditional back-matching strategy demonstrated that the ordinal is a capable metric for stereo correspondence. However, the traditional back-matching strategy was computationally complex with at least $\Theta(MNDU)$ growth to support the search range D and reverse matching depth U. Also, the traditional matching strategy had varying memory requirements and required at least three times the memory buffering of the original stereo pair to support the Gaussian pyramid and full-size intermediate disparity solutions.

The ordinal was then investigated for performance with a more recent window-based two-dimensional, two-stage dynamic programming method. The dynamic programming produced raw results of much higher accuracy compared to the traditional strategy un-aided by ad-hoc error correction. Also, the dynamic programming robustness offered the opportunity for computational complexity optimizations. Using these optimizations, we found that the dynamic programming method had a constant computational complexity of $\Theta(M_i N_i D)$ based on the M_i rows and N_i columns of the chosen sub-region partitioning and search range D. This computational complexity is clearly less than the complexity of the traditional methods. Additionally, the memory requirements were consistently proportional to the computational complexity since the dynamic programming algorithm uses no dynamic memory allocation like the traditional methods. Thus, the dynamic programming algorithm distinctly outperformed the traditional methods in accuracy, complexity and memory requirements.

Second, the ordinal was investigated for its candidacy toward hardware implementation. The proposed architecture can feasibly be synthesized in an FPGA and theoretically computes the first correlation coefficient κ in $5+\lceil \log_2 n \rceil$ clock cycles at 100 MHz in an Altera Cyclone family FPGA, fully pipelined. While an actual implementation was not possible for this research due to time constraints, an implementation of only the ordinal as a macro function with the search and solver (i.e., the investigated dynamic programming algorithm) left to software could be sufficient to achieve real-time dense disparity map computation. This is because, for all experiments, computation of the correlation coefficient κ accounted for the majority of the processing time.

B. FUTURE WORK

This thesis presented a powerful method of solving the stereo correspondence problem using a logical metric and low complexity dynamic programming search and solution algorithms. The continuation of this thread of research is to proceed to an efficient hardware implementation and real-time demonstration of stereo correspondence on a pair of video inputs. Also, research can be done to implement a real-time stereo processor to help navigate real world situations for robotics.

LIST OF REFERENCES

[1] J. Kriegman, E. Triendl, and T. O. Binford, "Stereo vision and navigation in buildings for mobile robots," *Robotics and Automation, IEEE Transactions on*, vol. 5, pp. 792–803, 1989.

[2] R. B. Perlow and B. D. Steinberg, "Automatic stereo processing of high resolution radar imagery," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 33, pp. 802–812, 1997.

[3] S. Negahdaripour, "Epipolar geometry of opti-acoustic stereo imaging," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, pp. 1776–1788, 2007.

[4] F. Wallner, R. Graf, and R. Dillmann, "Real-time map refinement by fusing sonar and active stereo-vision," in *Robotics and Automation, 1995. Proceedings, 1995 IEEE International Conference on*, vol. 3, pp. 2968–2973, 1995.

[5] S. Jin, J. Cho, X. D.Pham, K. M. Lee, S.-K.Park, M.Kim, and J. W. Jeon, "FPGA design and implementation of a real-time stereo vision system," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 20, pp. 15–26, 2010.

[6] S. Baase, "Computer algorithms: introduction to design and analysis," *Menlo Park, CA: Addison-Wesley Publishing Company*, pp. 28–35, 1988.

[7] L. P. Yaroslavsky and V. Kober, "Redundancy of signals and transformations and computational complexity of signal and image processing," in *Pattern Recognition, 1994. Vol. 3— Conference C: Signal Processing, Proceedings of the 12th IAPR International Conference on*, pp. 164–166 vol. 3, 1994.

[8] R. Singh and A. Ortega, "Reduced-complexity delayed-decision algorithm for context-based image processing systems," *Image Processing, IEEE Transactions on*, vol. 16, pp. 1937–1945, 2007.

[9] A. Blake, "Stereo vision and segmentation," in *Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on*, pp. 4–4, 2007.

[10] D. Scharstein, R. Szeliski, and R. Zabih, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," in *Stereo and Multi-Baseline Vision, 2001. (SMBV 2001). Proceedings. IEEE Workshop on*, pp. 131–140, 2001.

[11] D. N. Bhat and S. K. Nayar, "Ordinal measures for image correspondence," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, pp. 415–423, 1998.

[12] C. Sun, “Rectangular subregioning and 3-D maximum-surface techniques for fast stereo matching,” in *Stereo and Multi-Baseline Vision, 2001. (SMBV 2001). Proceedings. IEEE Workshop on*, pp. 44–53, 2001.

[13] S. Sabihuddin, J. Islam, and W. J. MacLean, “Dynamic programming approach to high frame-rate stereo correspondence: A pipelined architecture implemented on a field programmable gate array,” in *Electrical and Computer Engineering, 2008. CCECE 2008. Canadian Conference on*, pp. 1461–1466, 2008.

[14] I.-H. Kim, D.-E. Kim, Y.-S. Cha, K.-H. Lee, and T.-Y. Kuc, “An embodiment of stereo vision system for mobile robot for real-time measuring distance and object tracking,” in *Control, Automation and Systems. ICCAS '07. International Conference on*, pp. 1029–1033, 2007.

[15] Y. Yan, Q. Zhu, Z. Lin, and Q. Chen, “Camera calibration in binocular stereo vision of moving robot,” in *Intelligent Control and Automation, 2006. WCICA 2006. the Sixth World Congress on*, pp. 9257–9261, 2006.

[16] Y. Ohta and T. Kanade, “Stereo by intra- and inter-scanline search using dynamic programming,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-7, pp. 139–154, 1985.

[17] D. N. Bhat and S. K. Nayar, “Ordinal measures for visual correspondence,” in *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR '96, 1996 IEEE Computer Society Conference on*, pp. 351–357, 1996.

[18] D. P. Huttenlocher and E. W. Jaquith, “Computing visual correspondence: Incorporating the probability of a false match,” in *Computer Vision, 1995. Proceedings, Fifth International Conference on*, pp. 515–522, 1995.

[19] CSIRO, Left stereo ball image, http://extra.cmis.csiro.au/IA/changs/stereo/ballL.gif (Accessed May 25, 2010).

[20] CSIRO, Right stereo ball image, http://extra.cmis.csiro.au/IA/changs/stereo/ballR.gif (Accessed May 25, 2010).

[21] J. Qian and J. Su, “Online estimation of image jacobian matrix by kalman-bucy filter for uncalibrated stereo vision feedback,” in *Robotics and Automation, 2002. Proceedings, ICRA '02. IEEE International Conference on*, vol. 1, pp. 562–567, 2002.

[22] Z. Zhang and O. Faugeras, “Building a 3D world model with a mobile robot: 3D line segment representation and integration,” in *Pattern Recognition, 1990. Proceedings, 10th International Conference on*, vol. 1, pp. 38–42, 1990.

[23] Stanford AI Lab, Left stereo meter image, http://ai.stanford.edu/~birch/p2p/meter1.jpg (Accessed May 25, 2010).

[24] Stanford AI Lab, Left stereo shrub image, http://ai.stanford.edu/~birch/p2p/shrub03.jpg (Accessed May 25, 2010).

[25] O. Veksler, “Stereo correspondence by dynamic programming on a tree,” in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, pp. 384–390, 2005.

[26] Stanford AI Lab, Right stereo meter image, http://ai.stanford.edu/~birch/p2p/meter2.jpg (Accessed May 25, 2010).

[27] Stanford AI Lab, Right stereo shrub image, http://ai.stanford.edu/~birch/p2p/shrub15.jpg (Accessed May 25, 2010).

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Chairman
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

4. Professor Robert Cristi
   Naval Postgraduate School
   Monterey, California

5. Professor Monique Fargues
   Naval Postgraduate School
   Monterey, California

6. ENS Bradley Ulis
   United States Navy
   Monterey, California